(12) UK Patent Application (19) GB (11) 2 232 514 (13) A

(43) Date of A publication 12.12.1990

(21) Application No 9006947.7

(22) Date of filing 28.03.1990

(30) Priority data
(31) 01104052 (32) 24.04.1989 (33) JP
01129707 23.05.1989
01275512 23.10.1989
01275513 23.10.1989

(51) INT CL[5]
G06F 9/44

(52) UK CL (Edition K)
G4A APX
U1S S2183

(56) Documents cited
GB 2177826 A GB 2028543 A EP 0138352 A2
EP 0134386 A2 EP 0120194 A2

(58) Field of search
UK CL (Edition K) G3N NGBA4 NGBD4, G4A APX
INT CL[5] G05B, G06F
On-line databases: WPI; INSPEC

(71) Applicant
Yokogawa Electric Corporation

(Incorporated in Japan)

9-32 Nakacho 2-chome, Musashino-shi, Tokyo 180, Japan

(72) Inventors
Masaaki Yonezawa
Kiyoshi Hasegawa
Yasunori Kawata
Kouji Matsuoka
Takasi Kadowaki

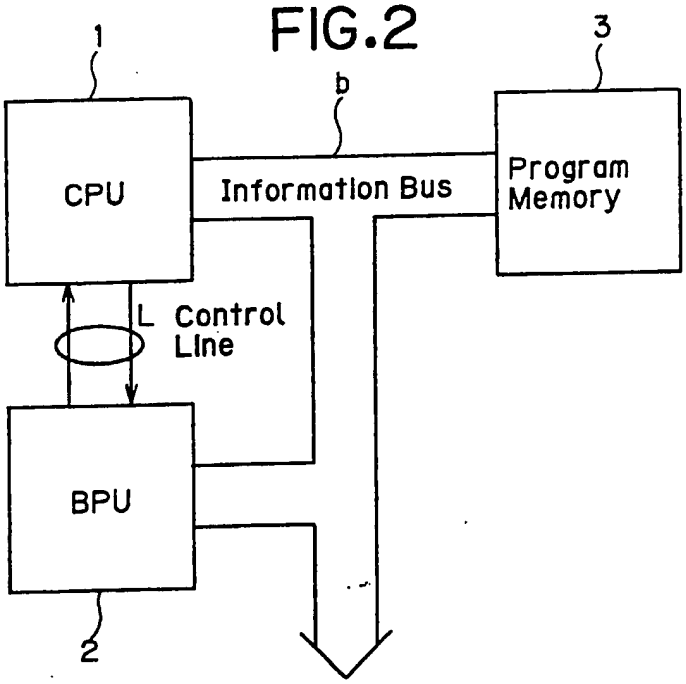
(74) Agent and/or Address for Service
Hughes Clark & Co
63 Lincoln's Inn Fields, London, WC2A 3JU, United Kingdom

(54) **Programmable controller**

(57) A programmable controller incorporating a sequence control program and adapted to input and output information to a variety of local intelligent appliances at a high velocity and efficiency in the field of factory automation for efficiently controlling factory production lines or of process automation for controlling multiple industrial processes comprises an improved change-over system associated with a 1-bit processor (BPU2) for executing the sequence control program and an ordinary processor (CPU1) directly connected thereto.

The controller is arranged such that when the processor which is executing the program determines that the command is to be executed by the other processor, it informs the other processor of the address in program memory from which execution is to start using a dedicated control line (L). The other processor reads the command directly from the program memory and executes it.

FIG.2



GB 2 232 514 A

FIG.1 (Prior Art)

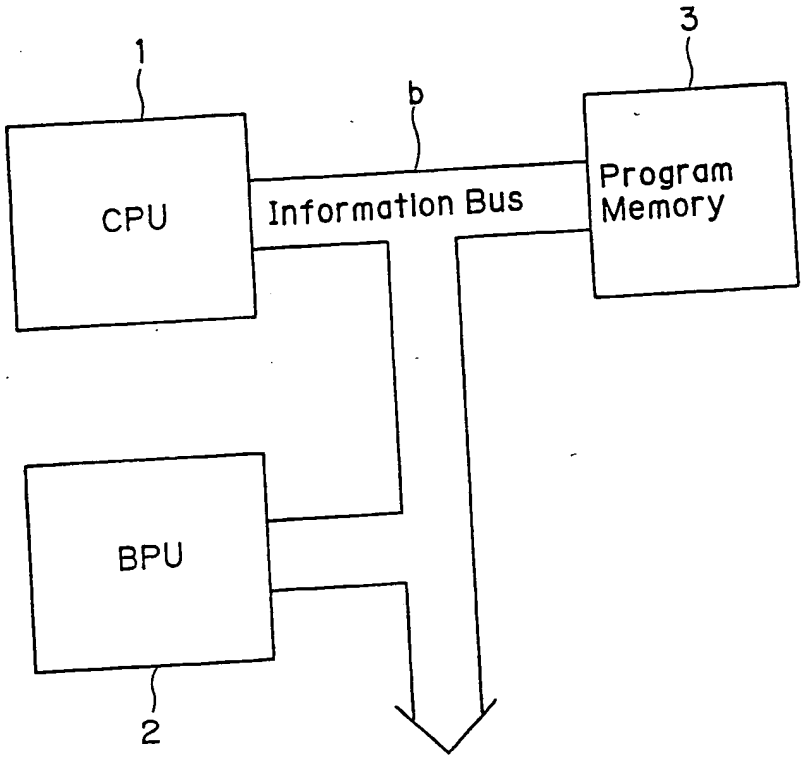


FIG.2

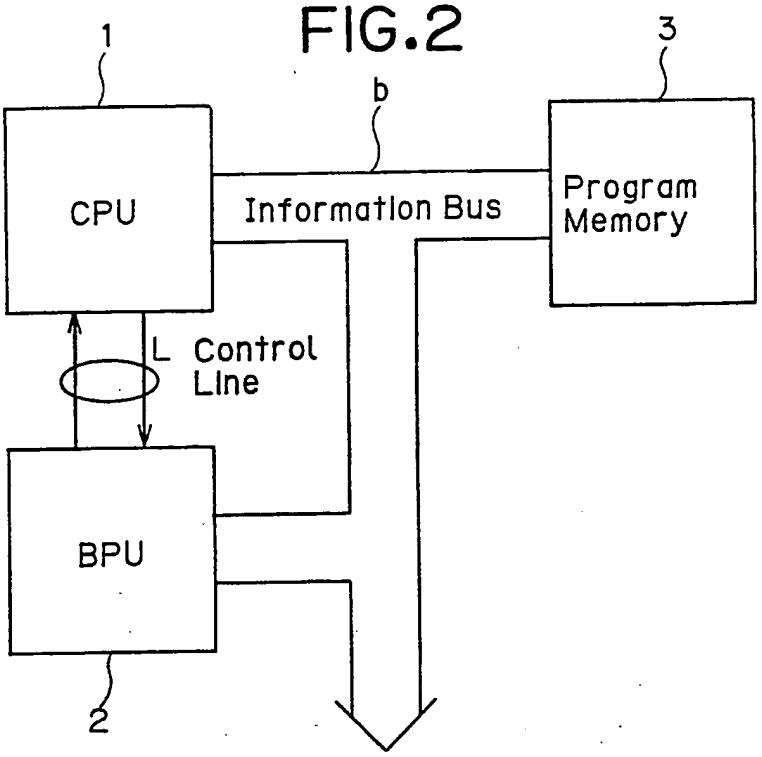

1
b
3
CPU
Information Bus
Program Memory
L Control Line
BPU
2

FIG.3

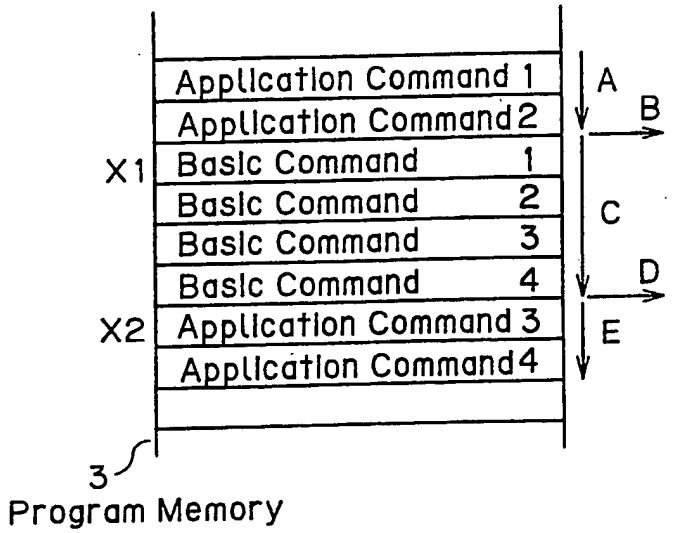

Application Command 1
Application Command 2
X1
Basic Command 1
Basic Command 2
Basic Command 3
Basic Command 4
X2
Application Command 3
Application Command 4
A
B
C
D
E
3
Program Memory

FIG.4

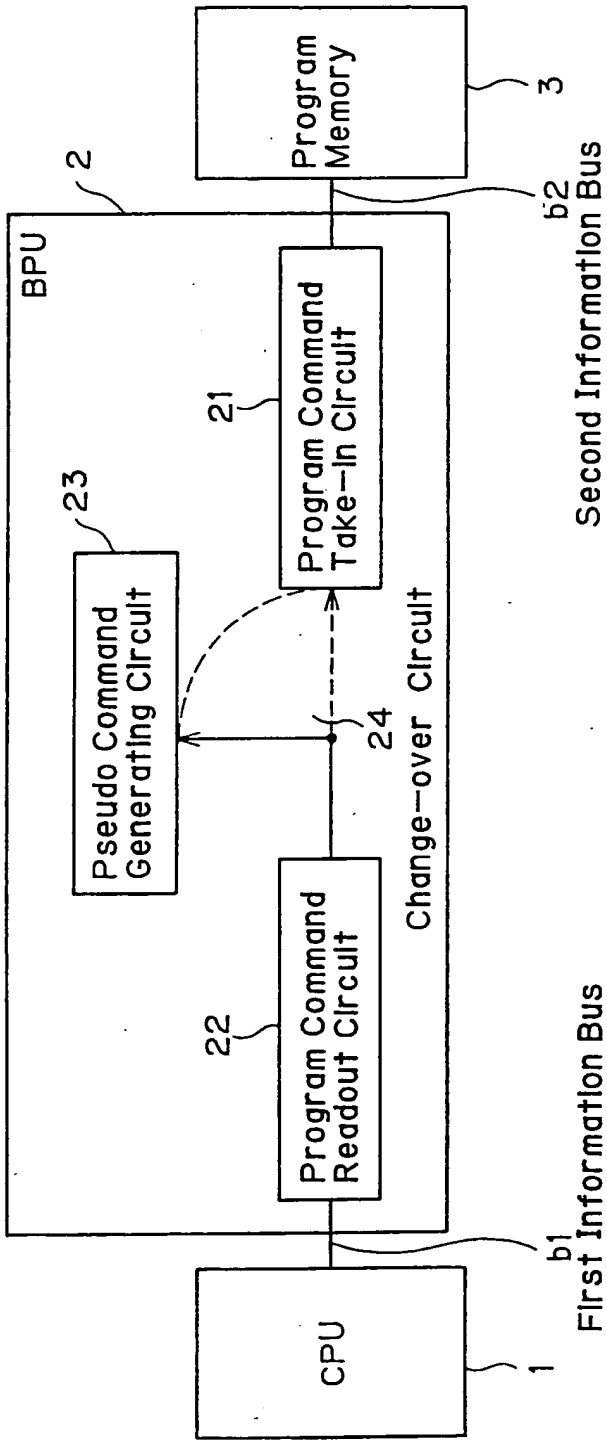

Program Memory
3
2
BPU
b2
Second Information Bus
Program Command Take-In Circuit
21
23
Pseudo Command Generating Circuit
24
Change-over Circuit
Program Command Readout Circuit
22
b1
First Information Bus
CPU
1

FIG.5(a)

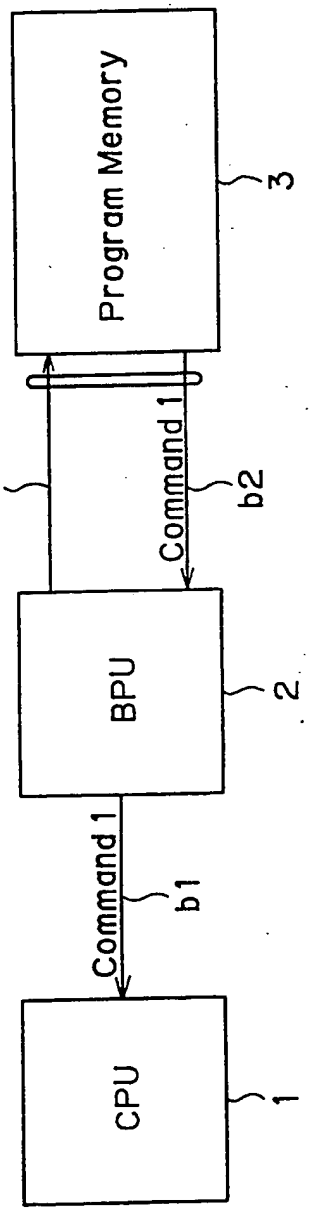

Program Memory
3
Y1
Command 1
b2
BPU
2
Command 1
b1
CPU
1

FIG.5(b)

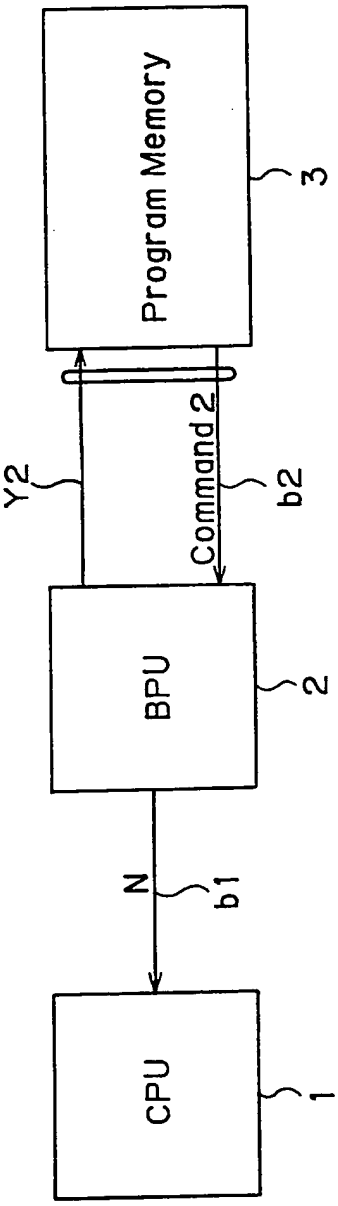

Program Memory
3
Y2
Command 2
b2
BPU
2
N
b1
CPU
1

FIG.6

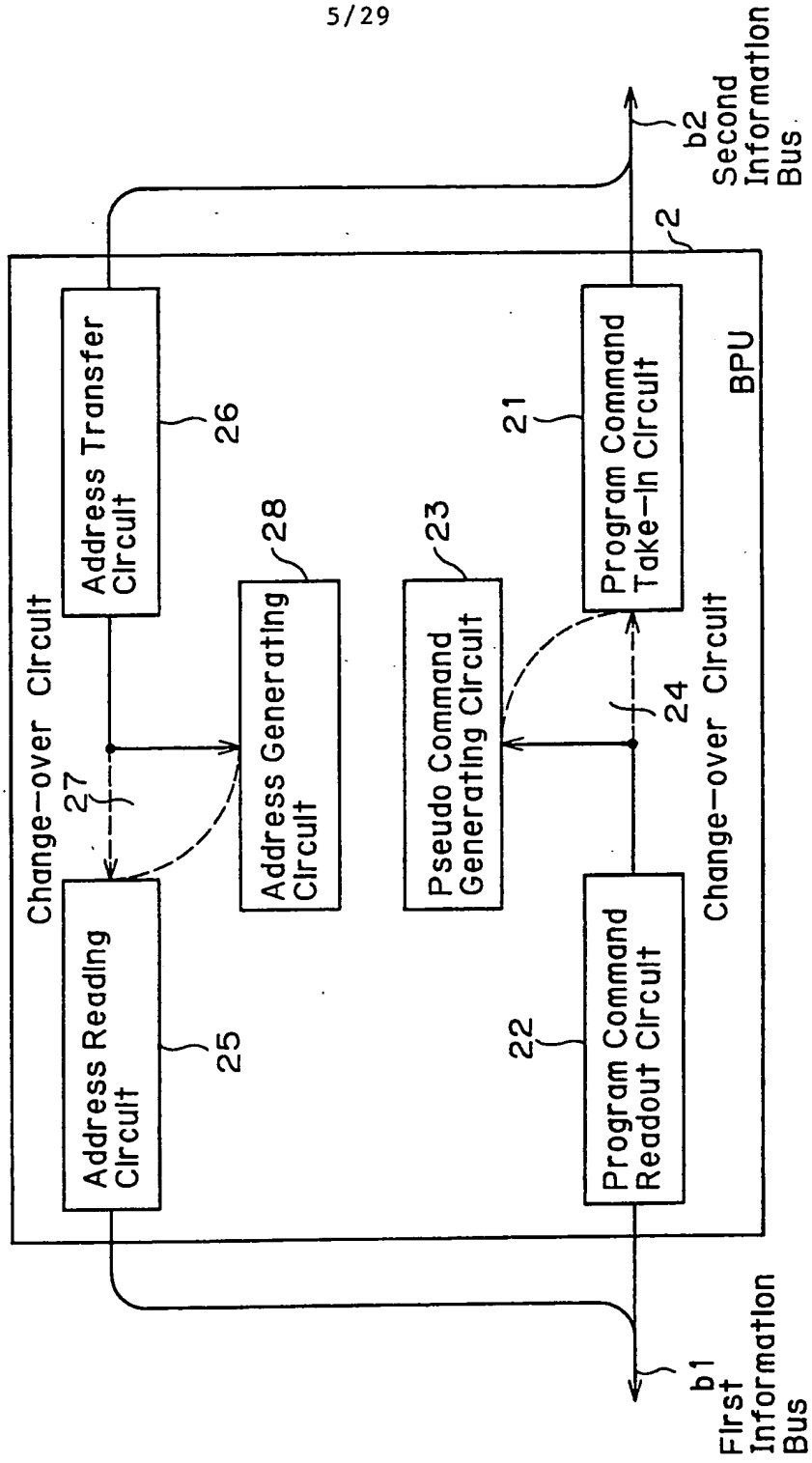

b2
Second
Information
Bus
2
BPU
Change-over Circuit
Address Transfer Circuit
26
Address Generating Circuit
28
Address Reading Circuit
25
27
Change-over Circuit
Program Command Take-In Circuit
21
Pseudo Command Generating Circuit
23
24
Program Command Readout Circuit
22
b1
First
Information
Bus

FIG.7(a)

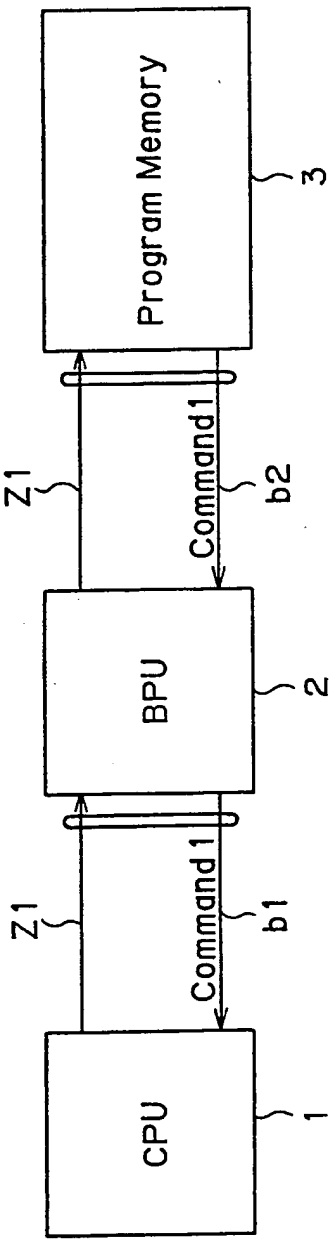

Program Memory
3
Z1
Command1
b2
BPU
2
Z1
Command1
b1
CPU
1

FIG.7(b)

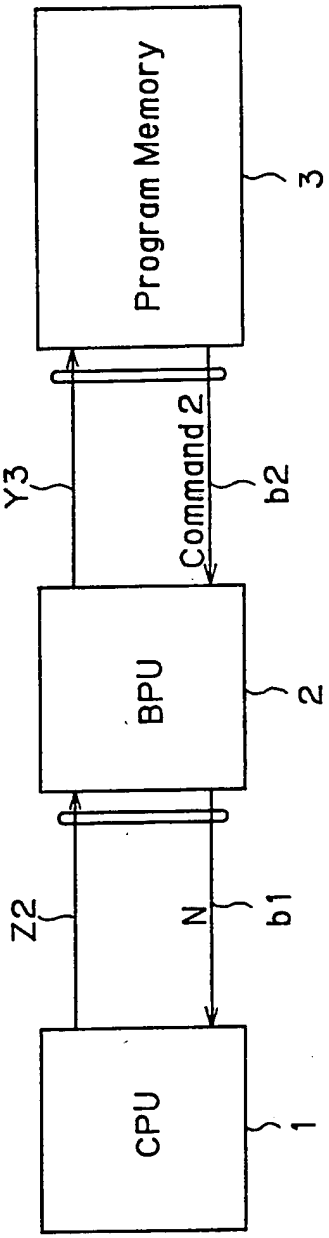

Program Memory
3
Y3
Command2
b2
BPU
2
Z2
N
b1
CPU
1

FIG.8

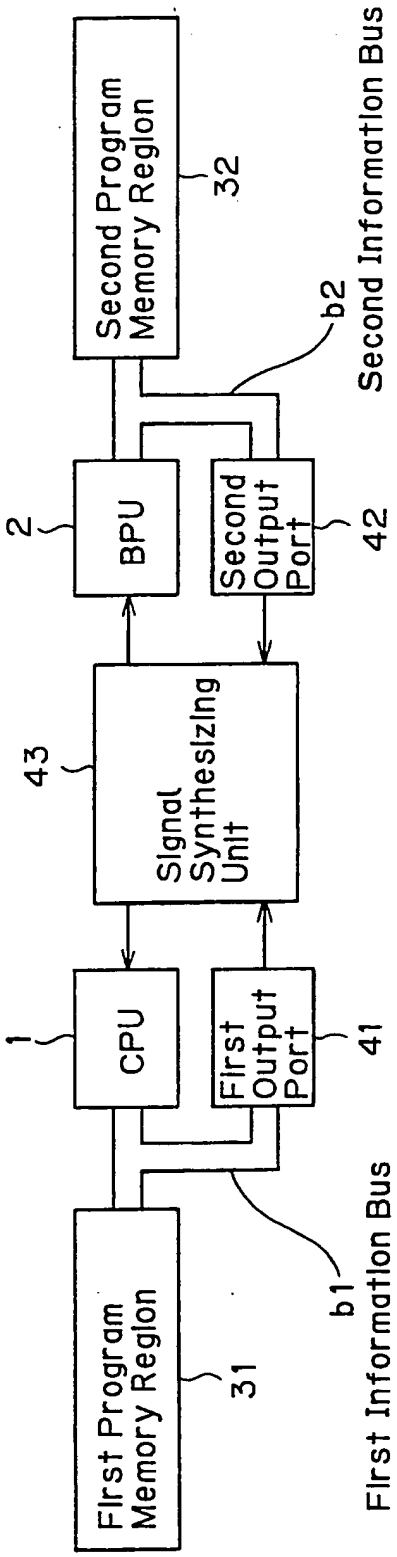

First Program Memory Region
31
First Information Bus
b1
CPU
1
First Output Port
41
Signal Synthesizing Unit
43
BPU
2
Second Output Port
42
b2
Second Information Bus
Second Program Memory Region
32

FIG.9(a)

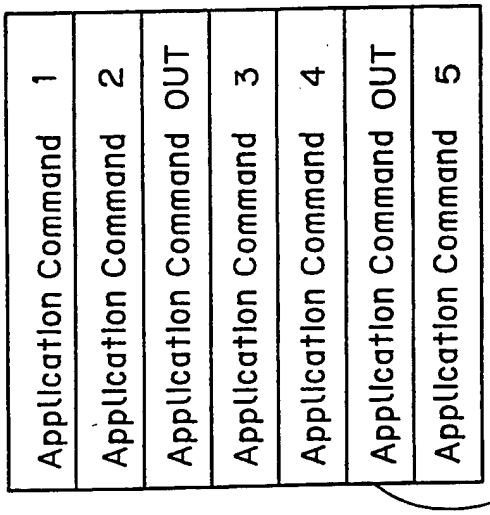

Application Command 1
Application Command 2
Application Command OUT
Application Command 3
Application Command 4
Application Command OUT
Application Command 5
31 First Program Memory Region

FIG.9(b)

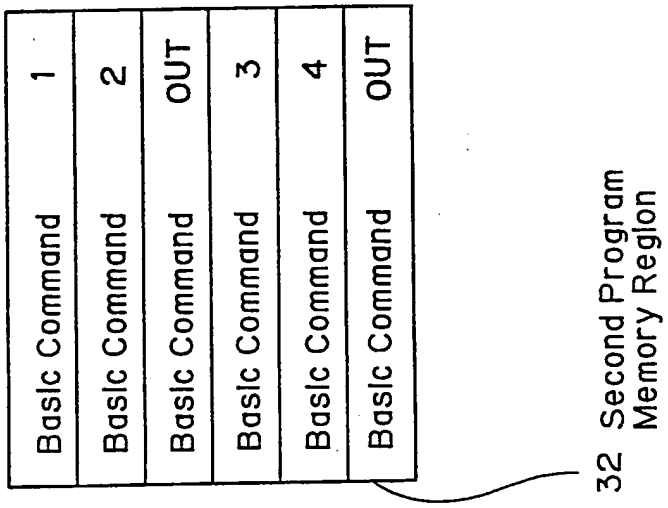

Basic Command 1
Basic Command 2
Basic Command OUT
Basic Command 3
Basic Command 4
Basic Command OUT
32 Second Program Memory Region

FIG.10(a)

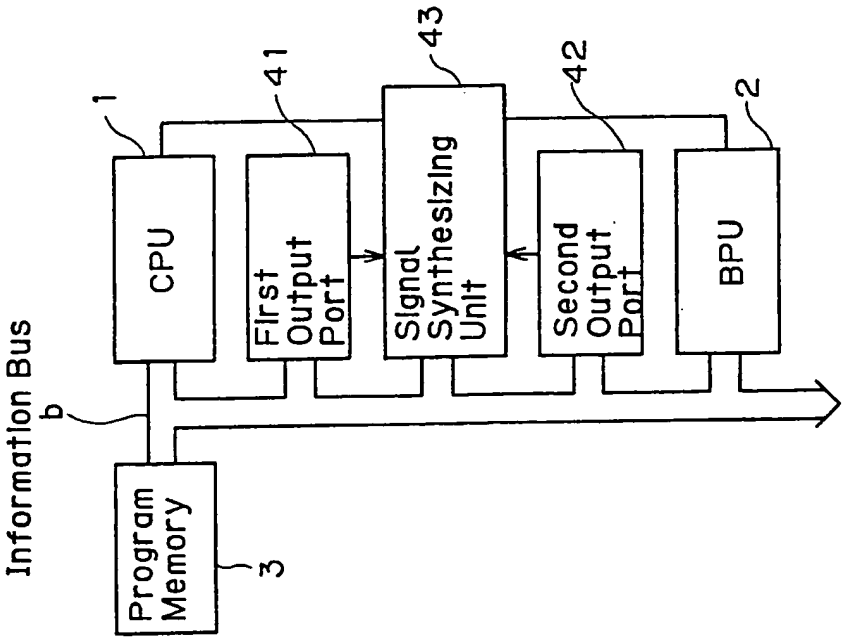


FIG.10(b)

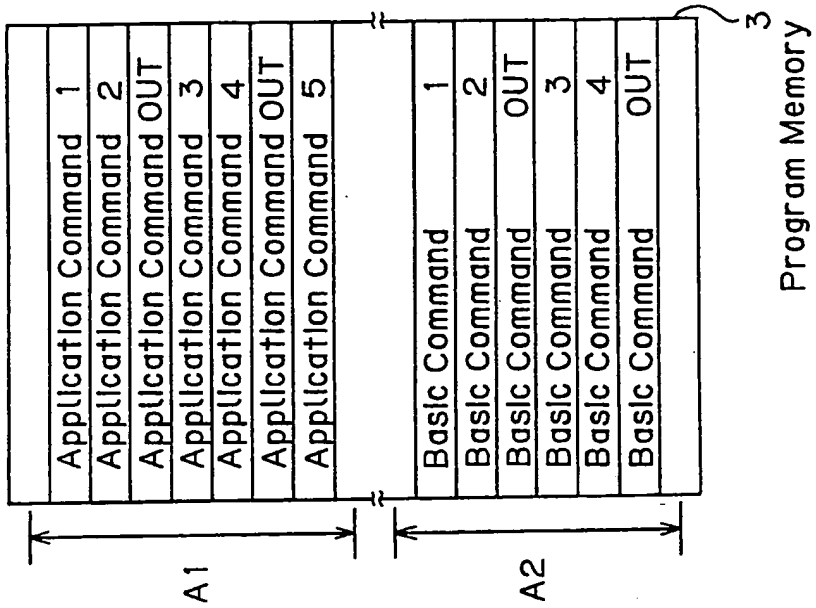


FIG.11

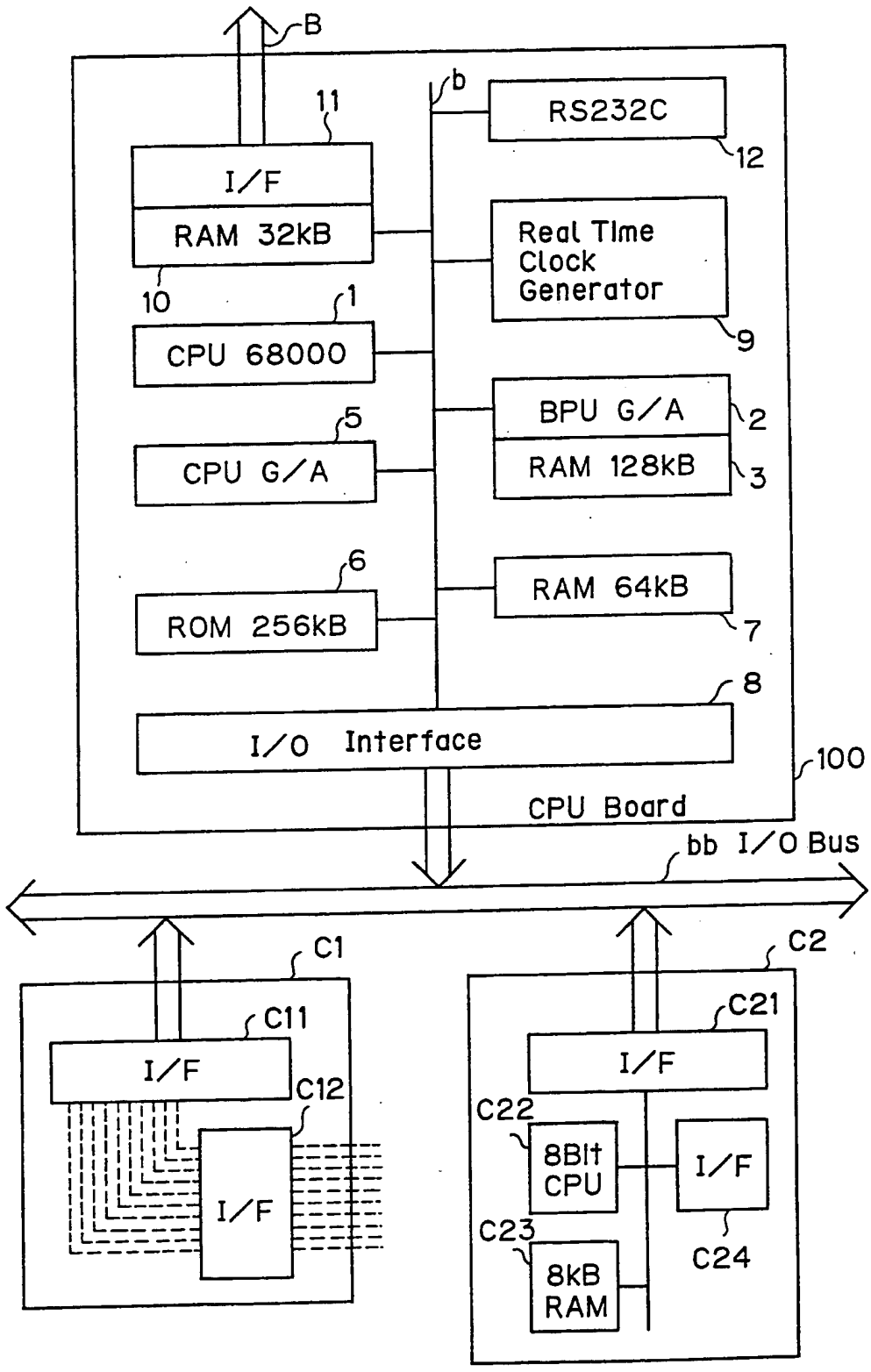

B
11
I/F
RAM 32kB
10
1
CPU 68000
5
CPU G/A
6
ROM 256kB
b
RS232C
12
Real Time Clock Generator
9
BPU G/A
2
RAM 128kB
3
RAM 64kB
7
8
I/O Interface
100
CPU Board
bb I/O Bus
C1
C11
I/F
C12
I/F
C2
C21
I/F
C22
8Bit CPU
I/F
C23
8kB RAM
C24

FIG.12

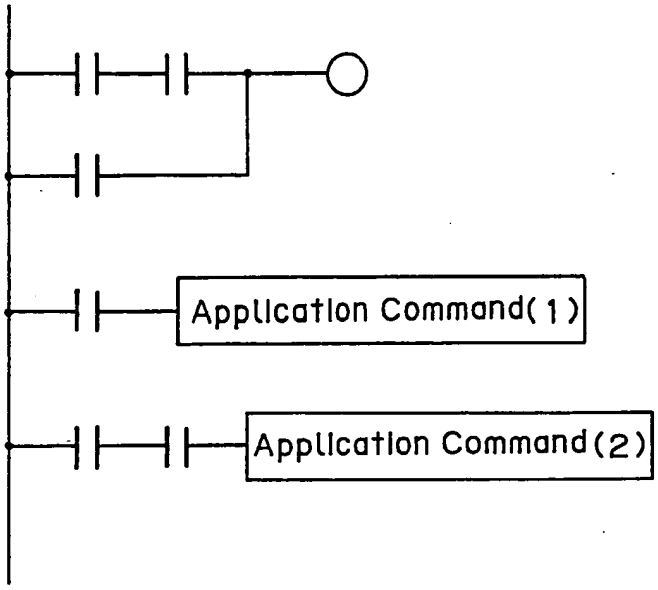

Application Command(1)
Application Command(2)

FIG.13

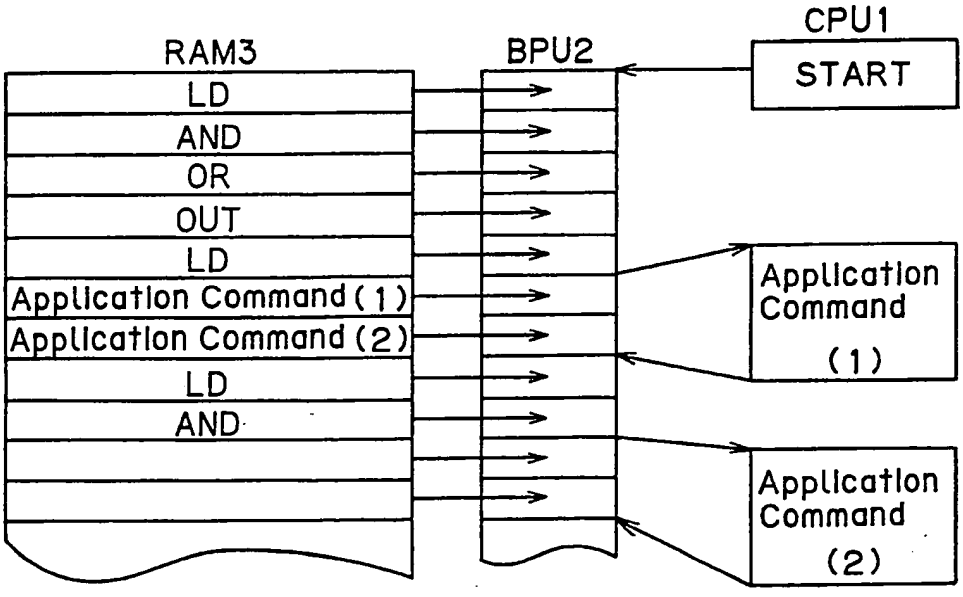

RAM3
BPU2
CPU1
START
LD
AND
OR
OUT
LD
Application Command(1)
Application Command(2)
LD
AND
Application Command (1)
Application Command (2)

FIG.14

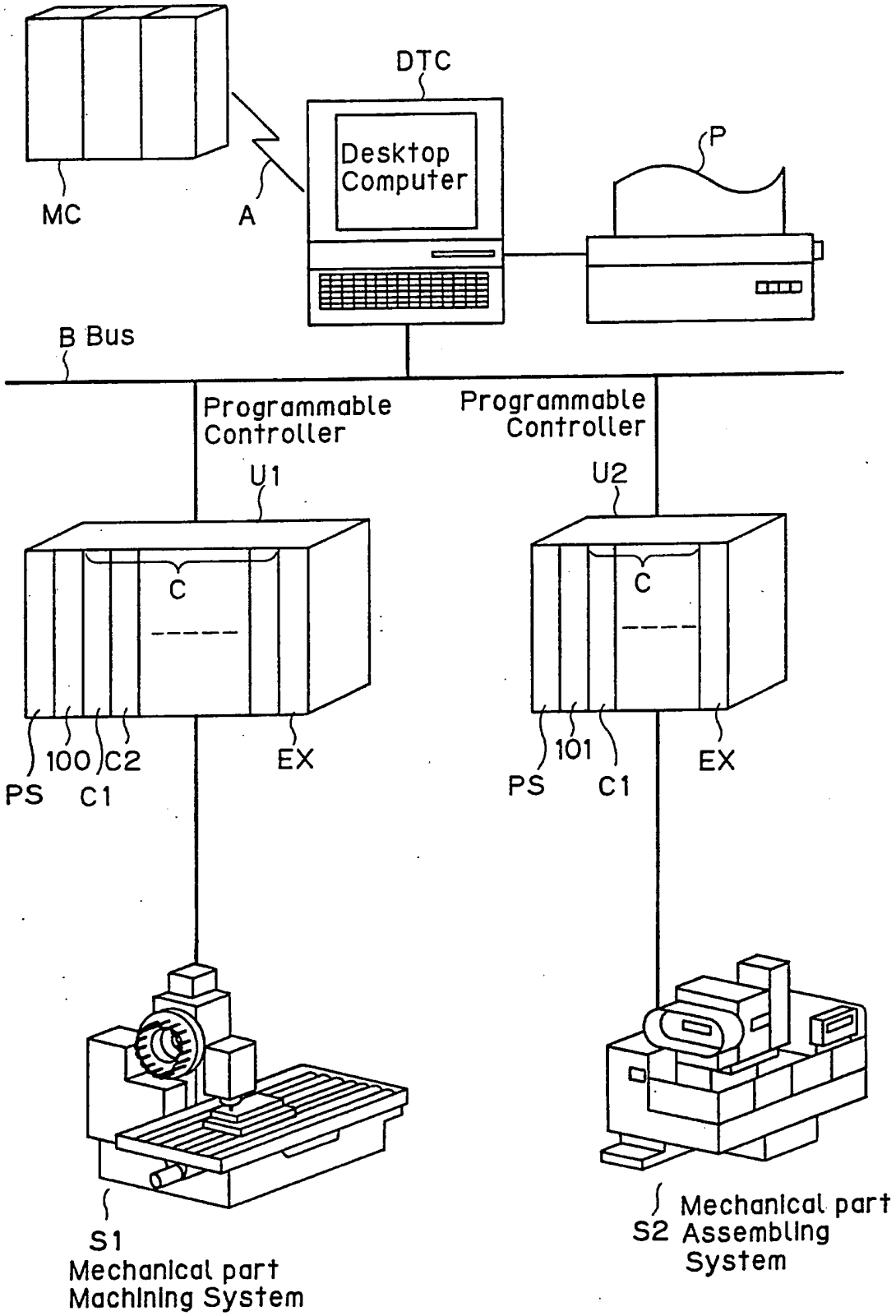

DTC
Desktop Computer
P
MC
A
B Bus
Programmable Controller
U1
Programmable Controller
U2
C
C
100
C2
PS
C1
EX
101
PS
C1
EX
S1
Mechanical part Machining System
S2
Mechanical part Assembling System

FIG.15

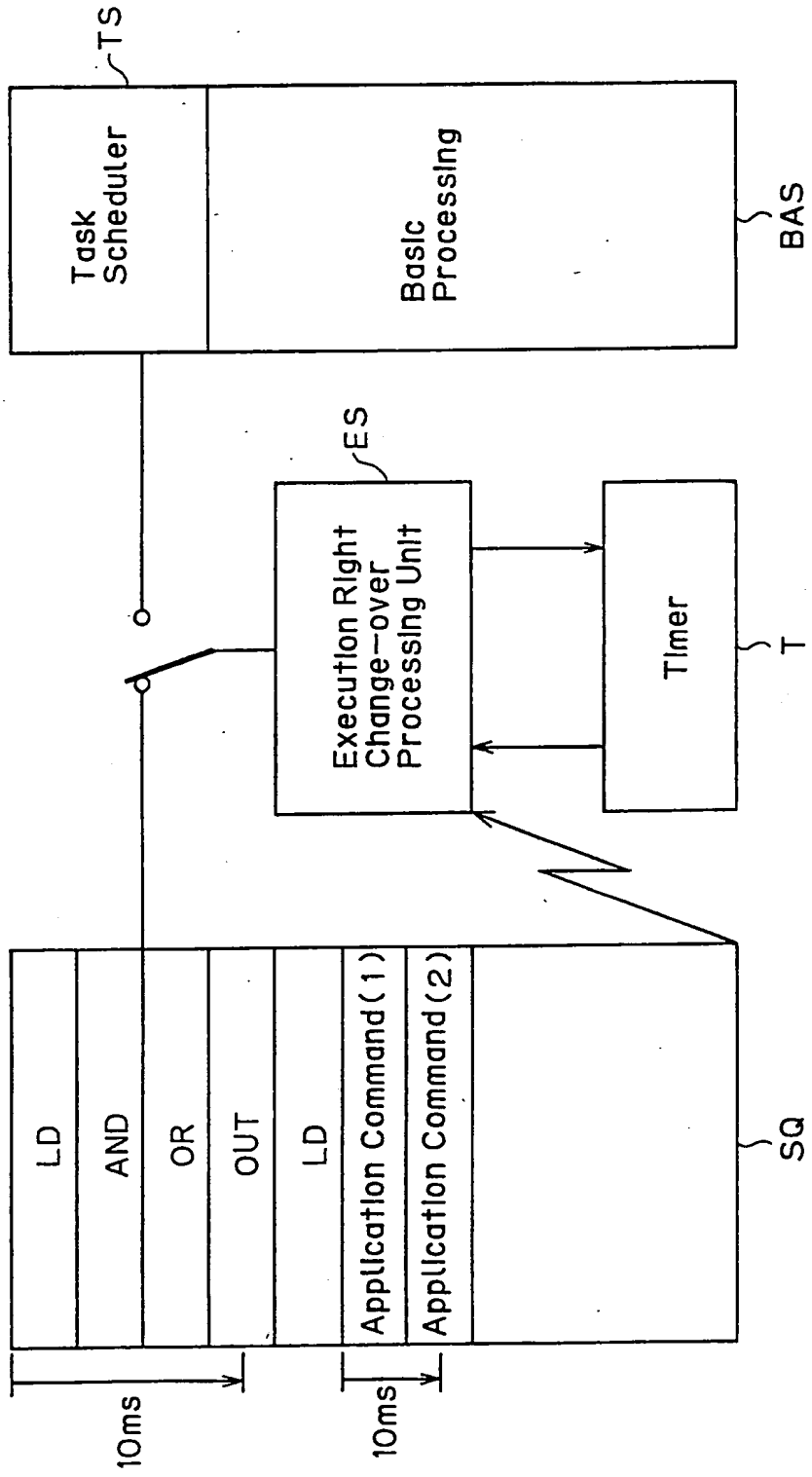

TS
Task Scheduler
Basic Processing
BAS
ES
Execution Right Change-over Processing Unit
Timer
T
LD
AND
OR
OUT
LD
Application Command (1)
Application Command (2)
SQ
10ms
10ms

FIG.16

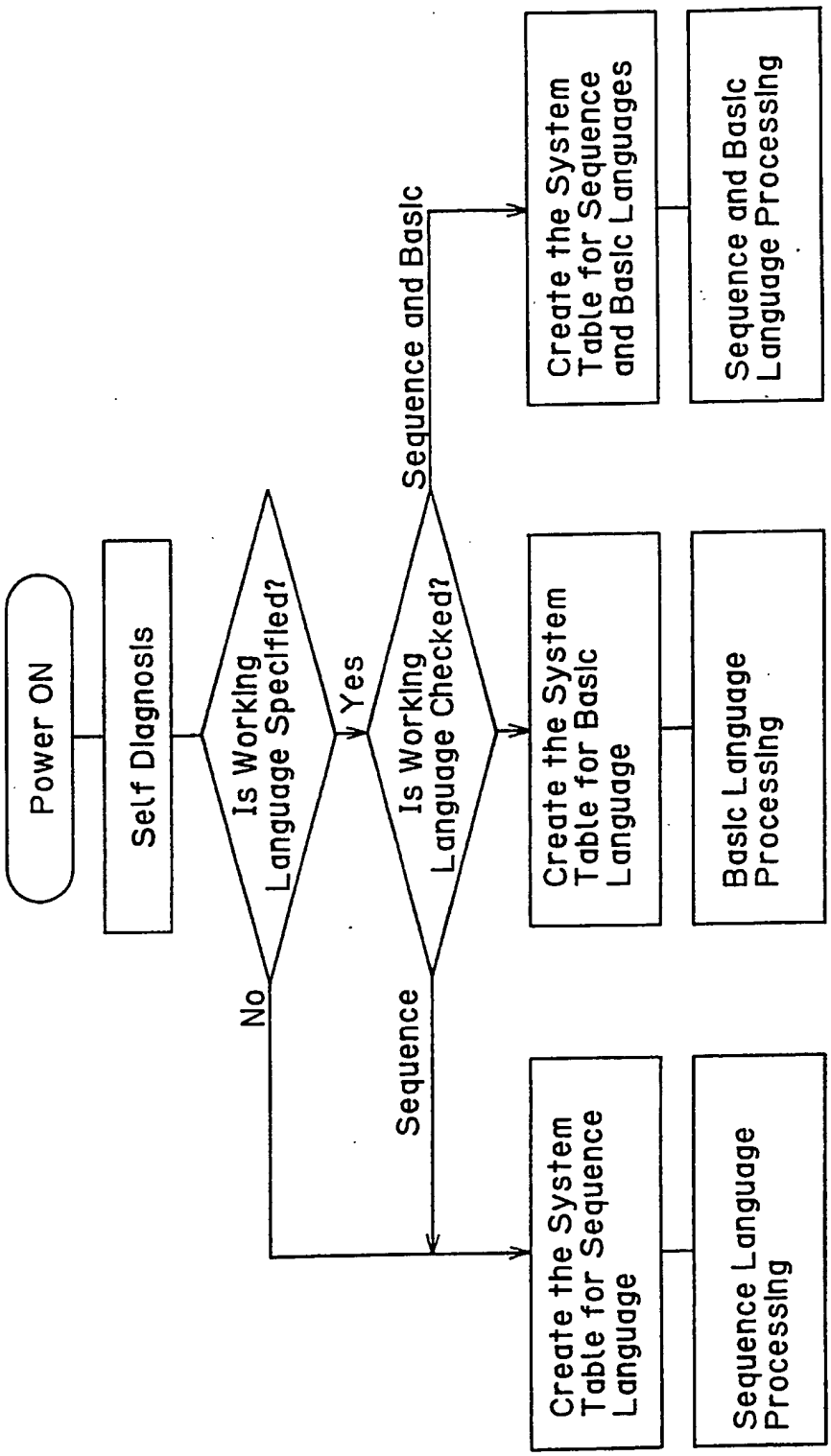

Power ON
Self Diagnosis
Is Working Language Specified?
No
Yes
Is Working Language Checked?
Sequence
Sequence and Basic
Create the System Table for Sequence Language
Sequence Language Processing
Create the System Table for Basic Language
Basic Language Processing
Create the System Table for Sequence and Basic Languages
Sequence and Basic Language Processing

FIG.17

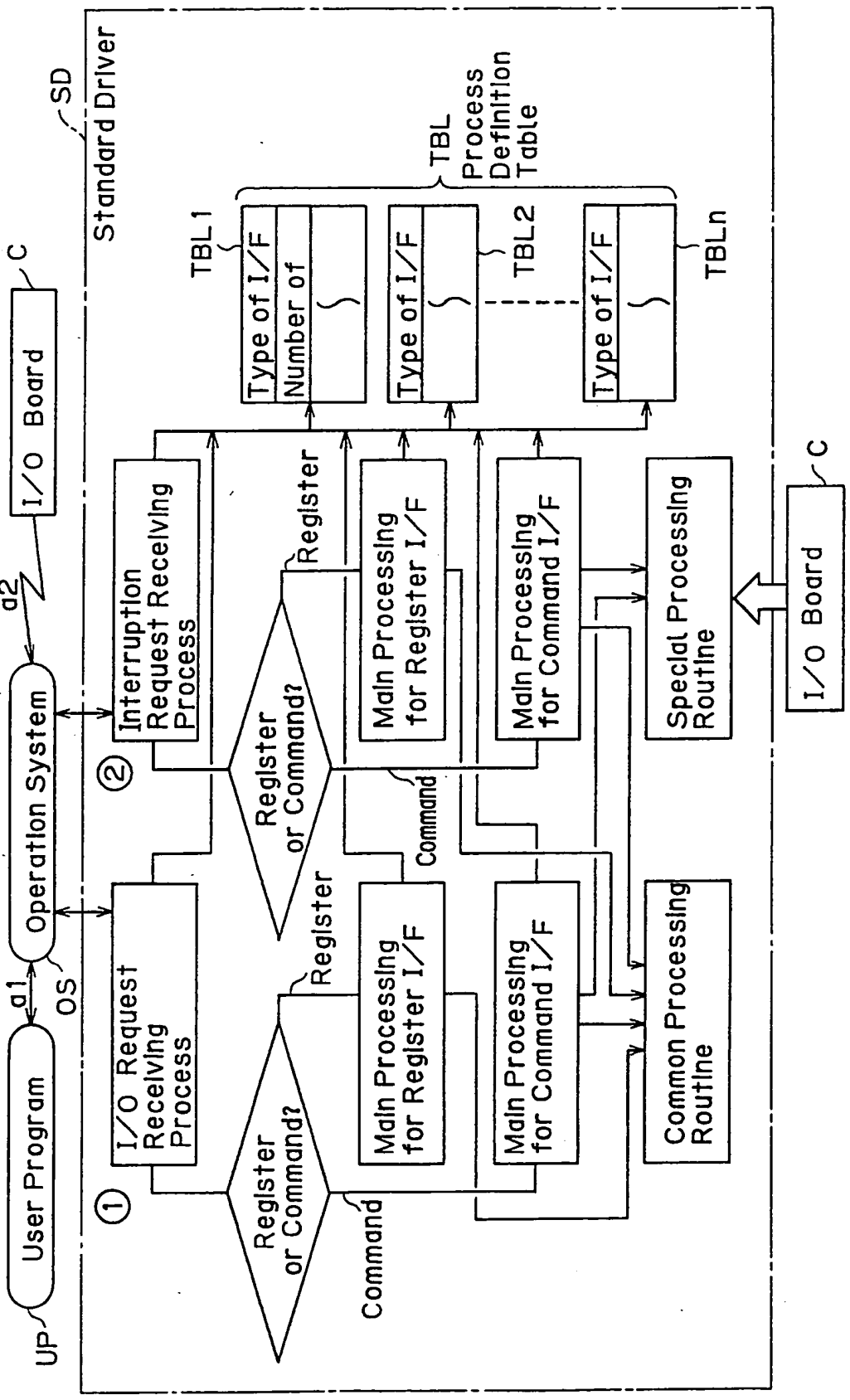

SD
Standard Driver
TBL
Process
Definition
Table
TBL1
TBL2
TBLn
Type of I/F
Number of
C
I/O Board
a2
Operation System
OS
a1
User Program
UP
I/O Request
Receiving
Process
Interruption
Request Receiving
Process
Register
or Command?
Register
Command
Main Processing
for Register I/F
Main Processing
for Command I/F
Special Processing
Routine
Common Processing
Routine
①
②

FIG.18

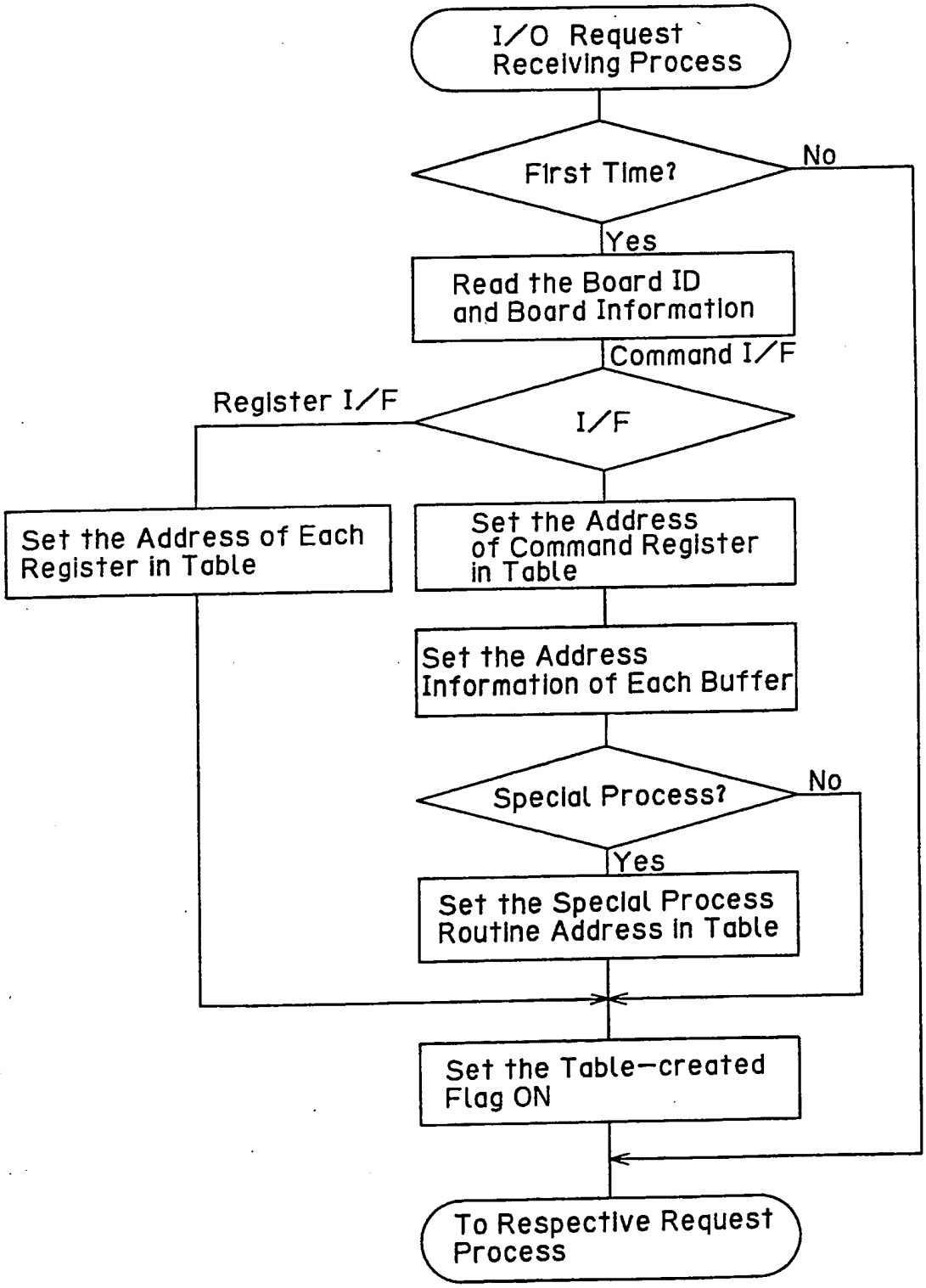

I/O Request Receiving Process
First Time?
No
Yes
Read the Board ID and Board Information
Command I/F
Register I/F
I/F
Set the Address of Each Register in Table
Set the Address of Command Register in Table
Set the Address Information of Each Buffer
Special Process?
No
Yes
Set the Special Process Routine Address in Table
Set the Table-created Flag ON
To Respective Request Process

FIG.19

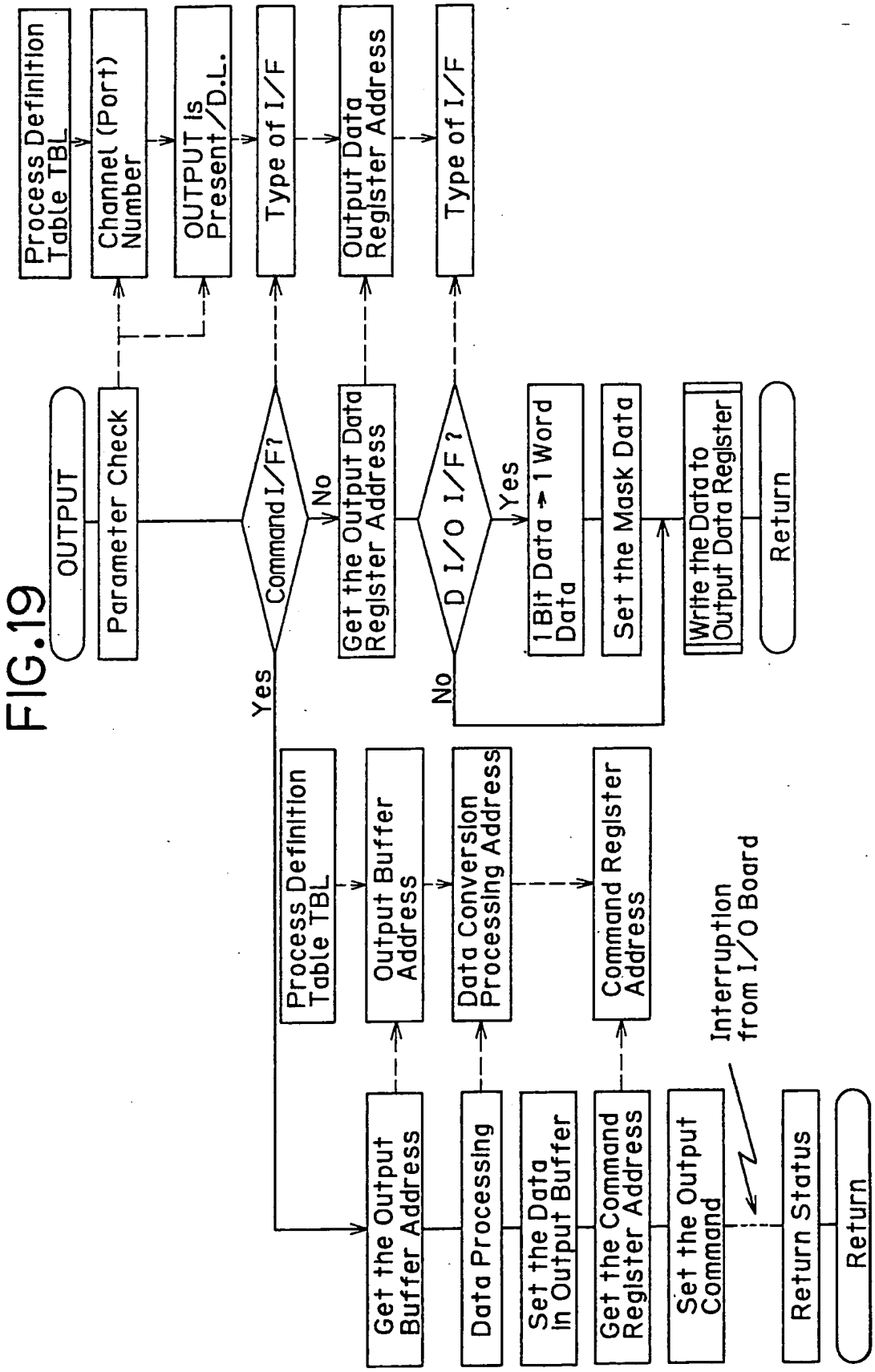

OUTPUT
Parameter Check
Process Definition Table TBL
Channel (Port) Number
OUTPUT is Present/D.L.
Command I/F?
Yes
No
Type of I/F
Output Data Register Address
Get the Output Data Register Address
D I/O I/F ?
No
Yes
Type of I/F
1 Bit Data → 1 Word Data
Set the Mask Data
Write the Data to Output Data Register
Return
Process Definition Table TBL
Output Buffer Address
Data Conversion Processing Address
Command Register Address
Interruption from I/O Board
Get the Output Buffer Address
Data Processing
Set the Data in Output Buffer
Get the Command Register Address
Set the Output Command
Return Status
Return

FIG.20

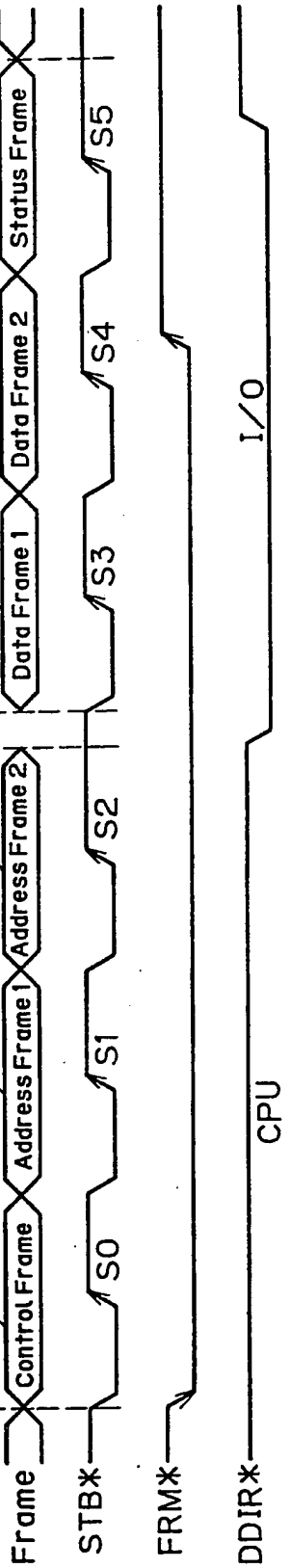

T1
T2
T3
F0
F1
F2
F3
F4
F5
Control Frame
Address Frame 1
Address Frame 2
Data Frame 1
Data Frame 2
Status Frame
S0
S1
S2
S3
S4
S5
Frame
STB*
FRM*
DDIR*
CPU
I/O

FIG.21

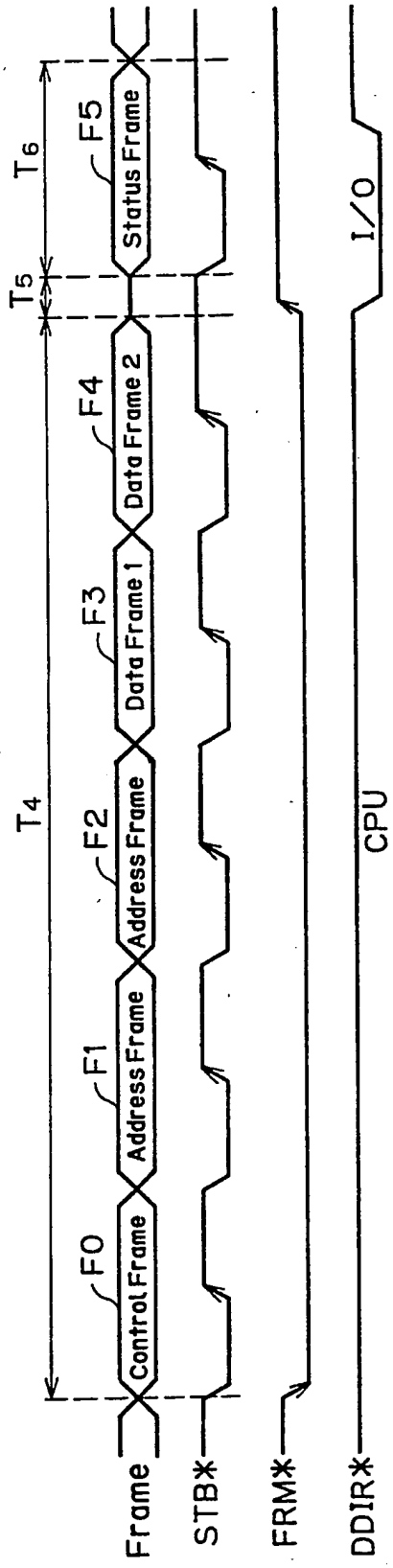

T4
T5
T6
F0
F1
F2
F3
F4
F5
Control Frame
Address Frame
Data Frame 1
Data Frame 2
Status Frame
Frame
STB*
FRM*
DDIR*
CPU
I/O

FIG.22

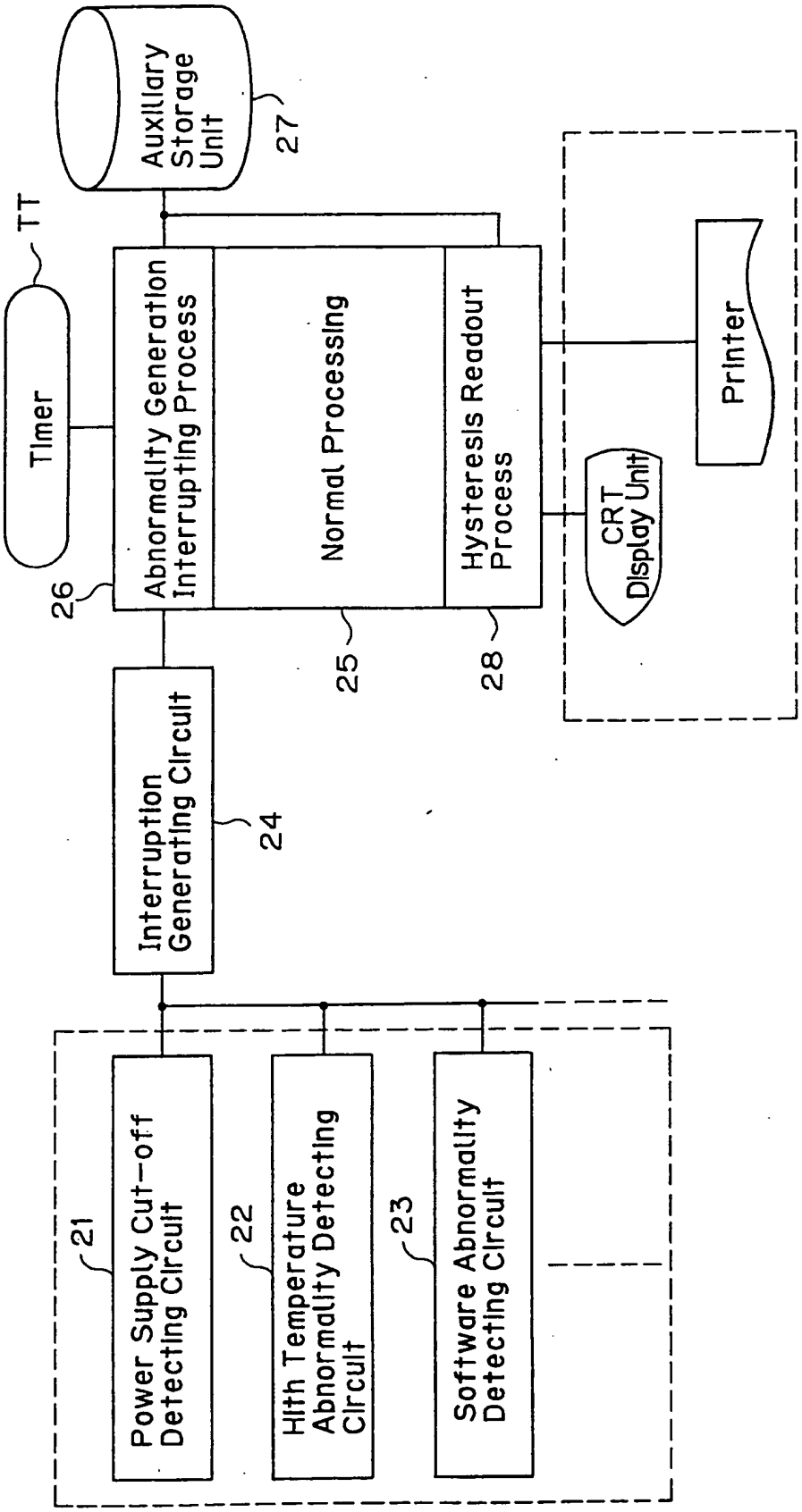

Auxiliary Storage Unit
27
TT
Timer
Abnormality Generation Interrupting Process
Normal Processing
Hysteresis Readout Process
26
25
28
CRT Display Unit
Printer
Interruption Generating Circuit
24
21
Power Supply Cut-off Detecting Circuit
22
Hith Temperature Abnormality Detecting Circuit
23
Software Abnormality Detecting Circuit

FIG.23

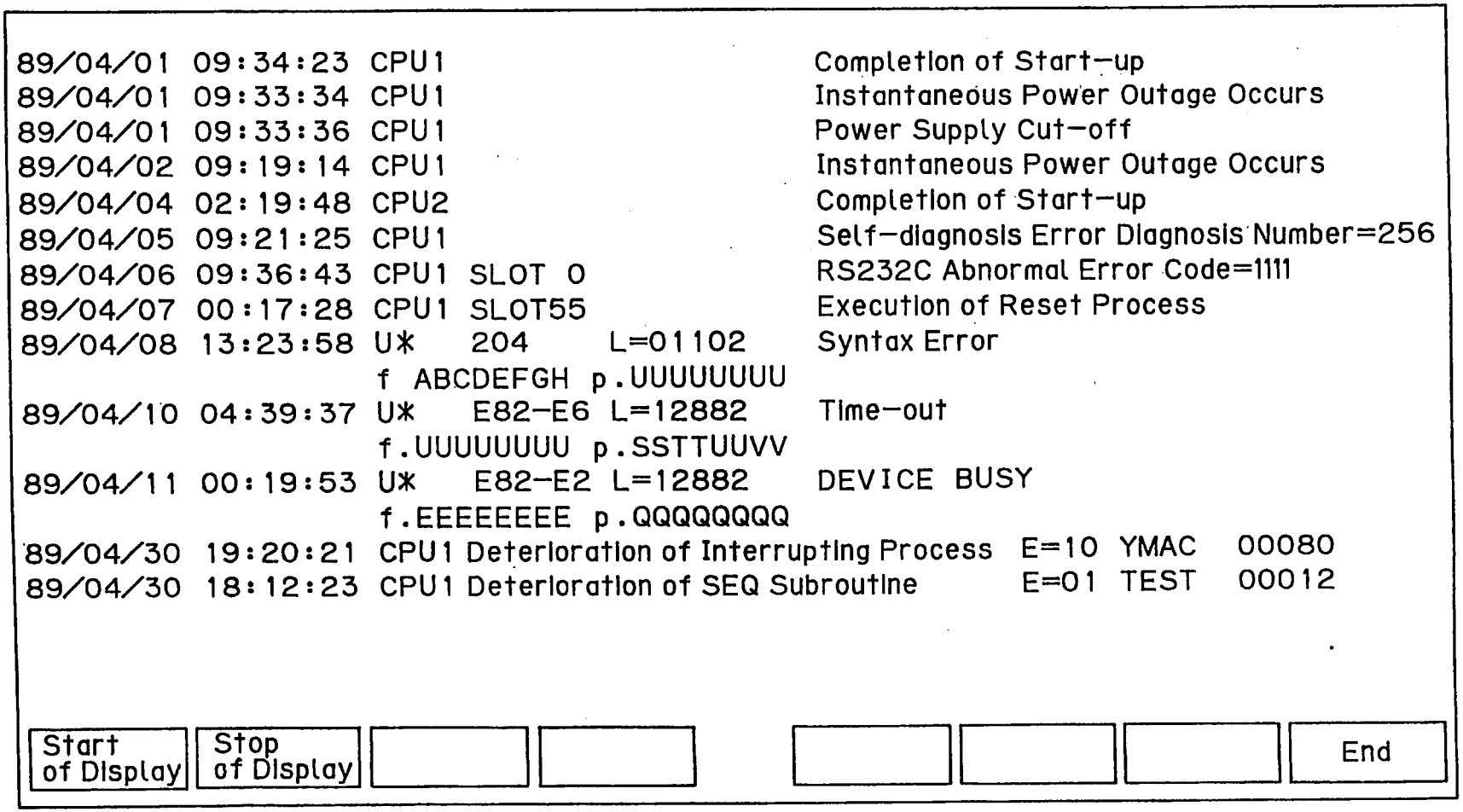

89/04/01 09:34:23 CPU1 Completion of Start-up
89/04/01 09:33:34 CPU1 Instantaneous Power Outage Occurs
89/04/01 09:33:36 CPU1 Power Supply Cut-off
89/04/02 09:19:14 CPU1 Instantaneous Power Outage Occurs
89/04/04 02:19:48 CPU2 Completion of Start-up
89/04/05 09:21:25 CPU1 Self-diagnosis Error Diagnosis Number=256
89/04/06 09:36:43 CPU1 SLOT 0 RS232C Abnormal Error Code=1111
89/04/07 00:17:28 CPU1 SLOT55 Execution of Reset Process
89/04/08 13:23:58 U* 204 L=01102 Syntax Error
f ABCDEFGH p.UUUUUUUU
89/04/10 04:39:37 U* E82-E6 L=12882 Time-out
f.UUUUUUUU p.SSTTUUVV
89/04/11 00:19:53 U* E82-E2 L=12882 DEVICE BUSY
f.EEEEEEEE p.QQQQQQQQ
89/04/30 19:20:21 CPU1 Deterioration of Interrupting Process E=10 YMAC 00080
89/04/30 18:12:23 CPU1 Deterioration of SEQ Subroutine E=01 TEST 00012
Start of Display
Stop of Display
End

FIG.24

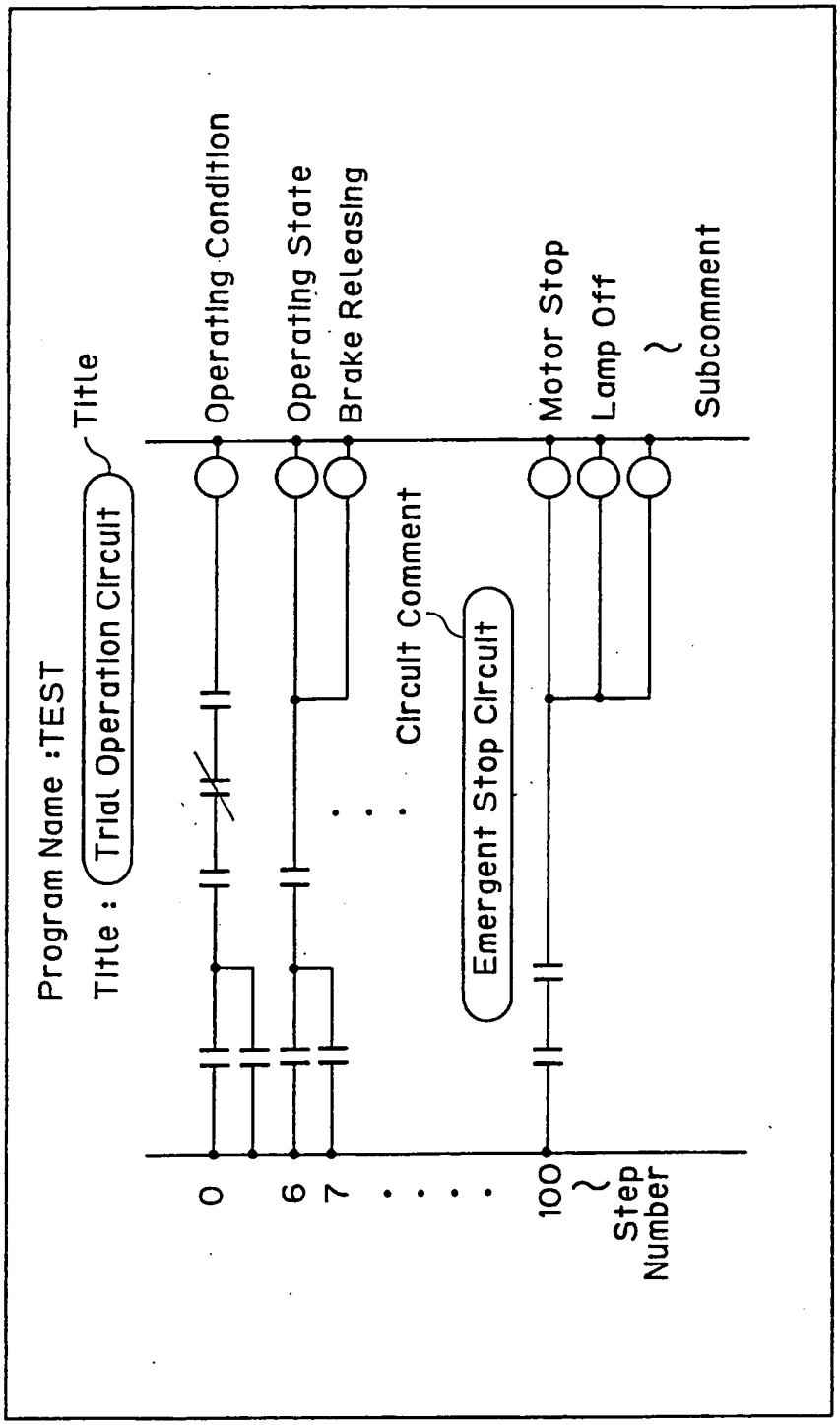

Program Name :TEST
Title :
Trial Operation Circuit
Title
Operating Condition
Operating State
Brake Releasing
Circuit Comment
Emergent Stop Circuit
Motor Stop
Lamp Off
Subcomment
0
6
7
100
Step Number

FIG.25

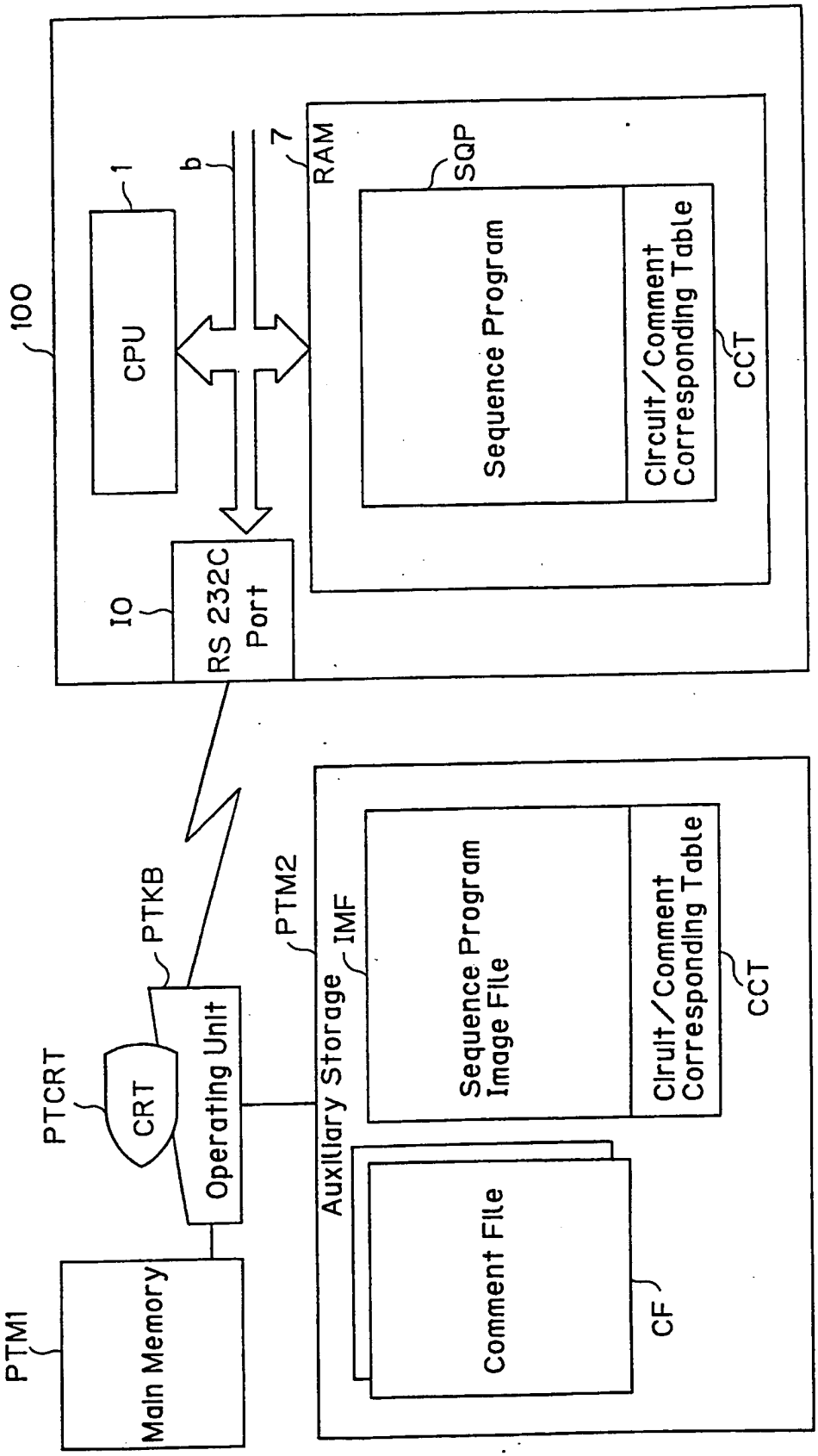

CPU Board
100
CPU
1
b
7
RAM
SQP
Sequence Program
Circuit/Comment
Corresponding Table
CCT
IO
RS 232C
Port
PTKB
PTCRT
CRT
Operating Unit
PTM2
IMF
Auxiliary Storage
Sequence Program
Image File
Circuit/Comment
Corresponding Table
CCT
Comment File
CF
PTM1
Main Memory

FIG.27

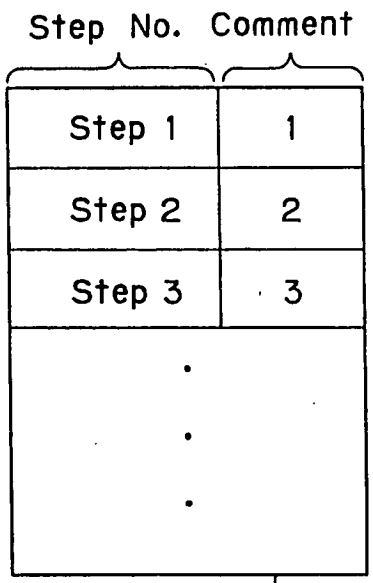

Step No.
Comment No.
Step 1
1
Step 2
2
Step 3
3
CCT
Circuit/Comment Corresponding Table

FIG.26

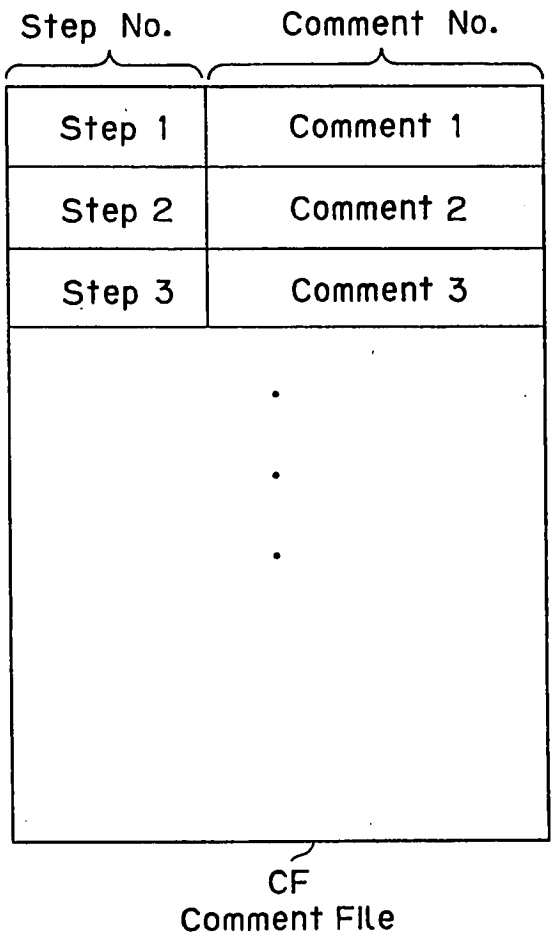

Step No.
Comment No.
Step 1
Comment 1
Step 2
Comment 2
Step 3
Comment 3
CF
Comment File

FIG.28

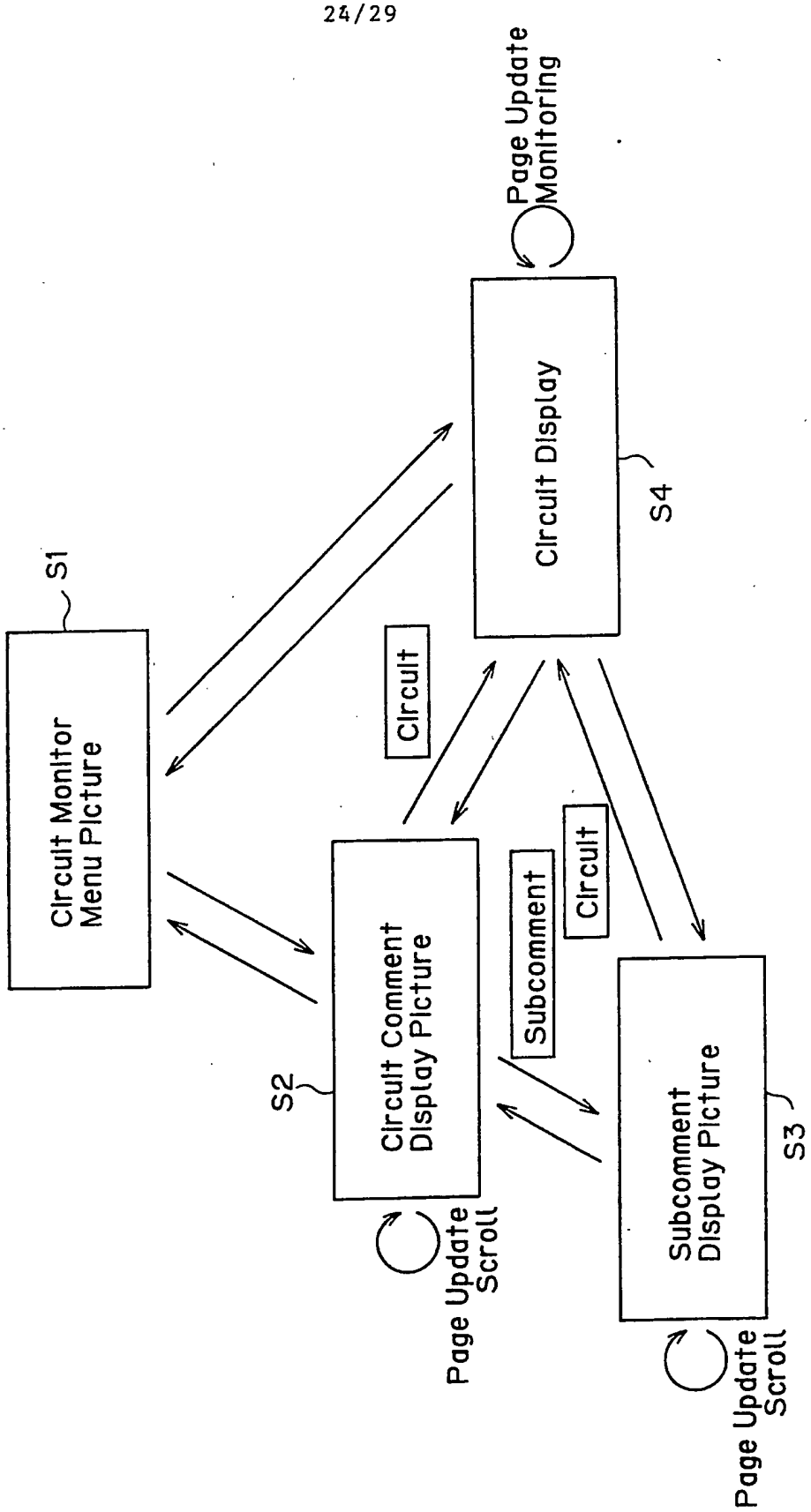

Circuit Monitor Menu Picture
S1
Circuit Display
S4
Page Update Monitoring
Circuit
Circuit Comment Display Picture
S2
Page Update Scroll
Subcomment
Circuit
Subcomment Display Picture
S3
Page Update Scroll

FIG.29

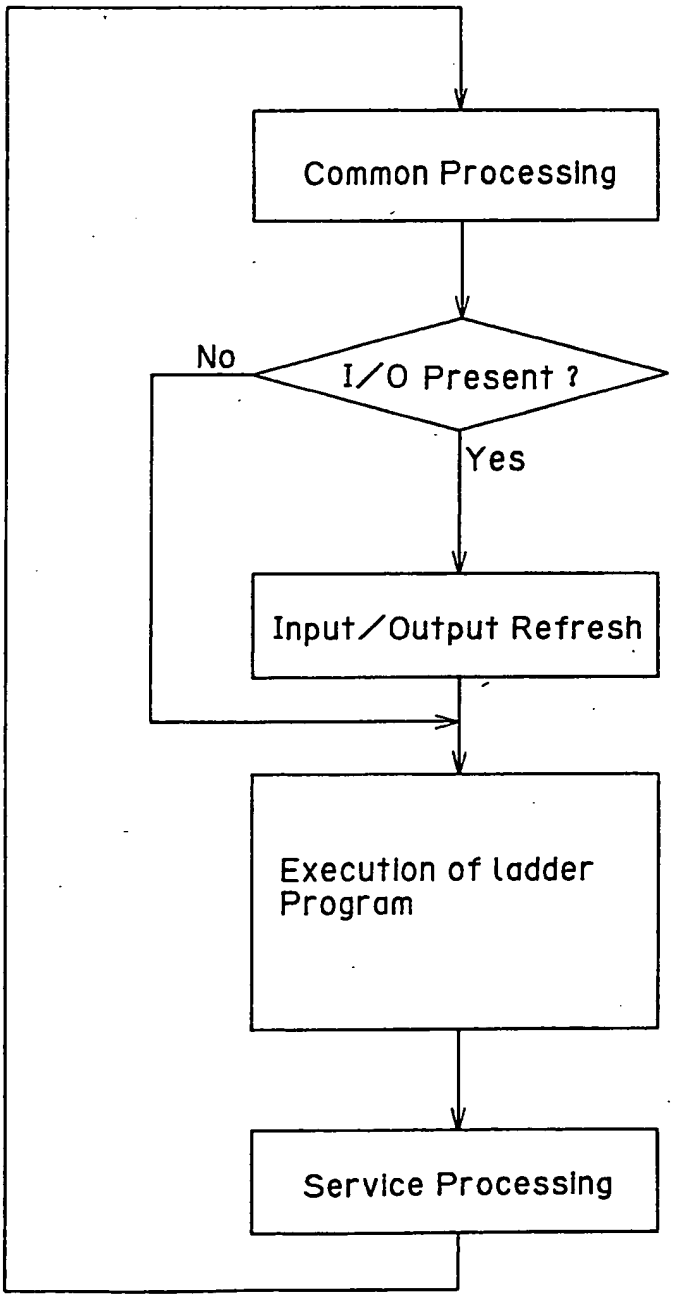

Common Processing
No
I/O Present ?
Yes
Input/Output Refresh
Execution of ladder Program
Service Processing

FIG.30

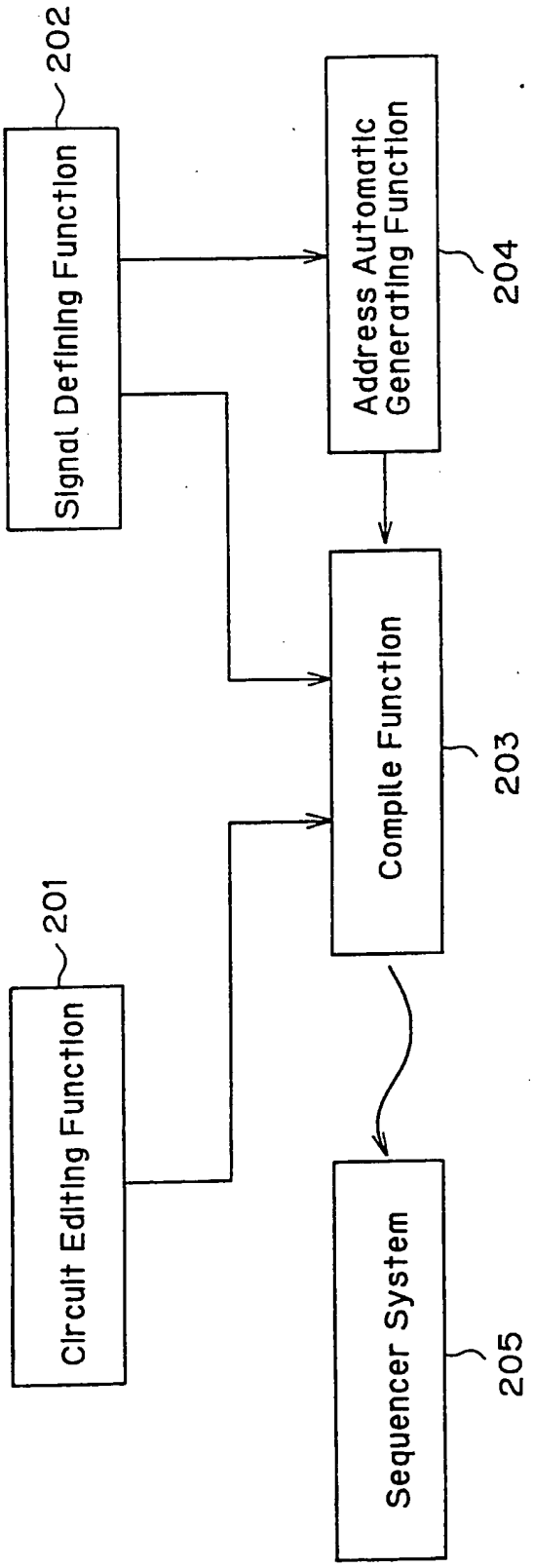

202
Signal Defining Function
Address Automatic Generating Function
204
Compile Function
203
201
Circuit Editing Function
Sequencer System
205

FIG.31

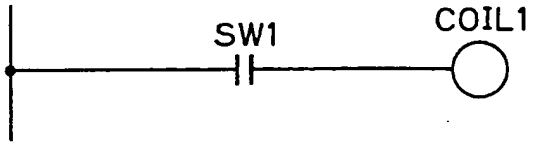


FIG.32

SW1 ⟶ X
SW2 ⟶ X
COIL1 ⟶ Y
COIL2 ⟶ Y
IRL1 ⟶ I
TIM1 ⟶ T
CNT1 ⟶ C
REG1 ⟶ D

FIG.33

SW1 ⟶ X001
SW2 ⟶ X002
COIL1 ⟶ Y001
COIL2 ⟶ Y002
IRL1 ⟶ I001
TIM1 ⟶ T001
CNT1 ⟶ C001
REG1 ⟶ D001

FIG.34(a)

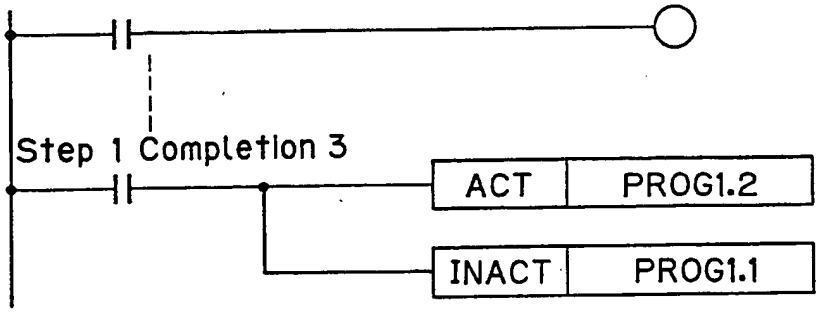


FIG.34(b)

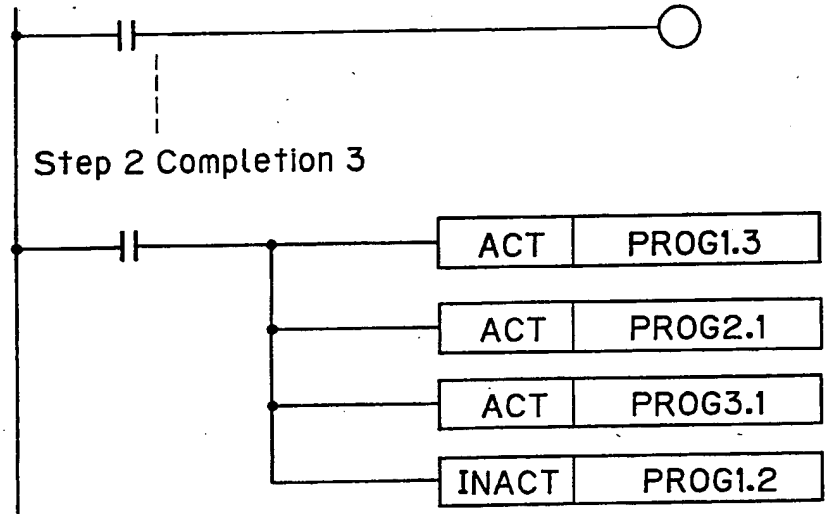


FIG.34(c)

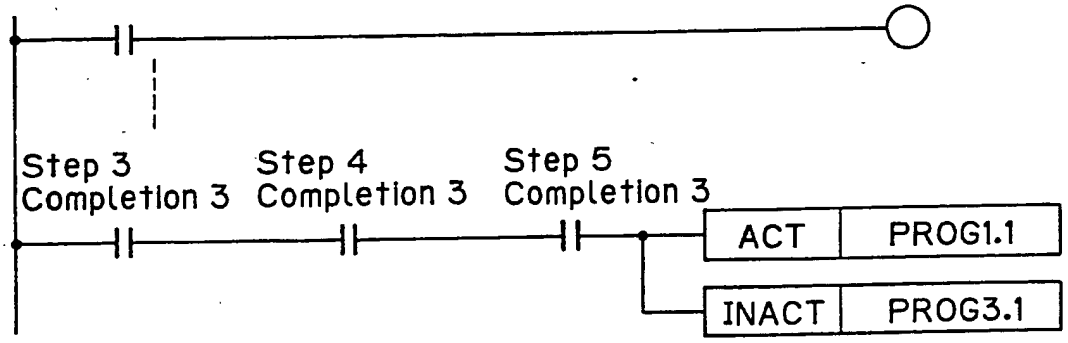


FIG.35

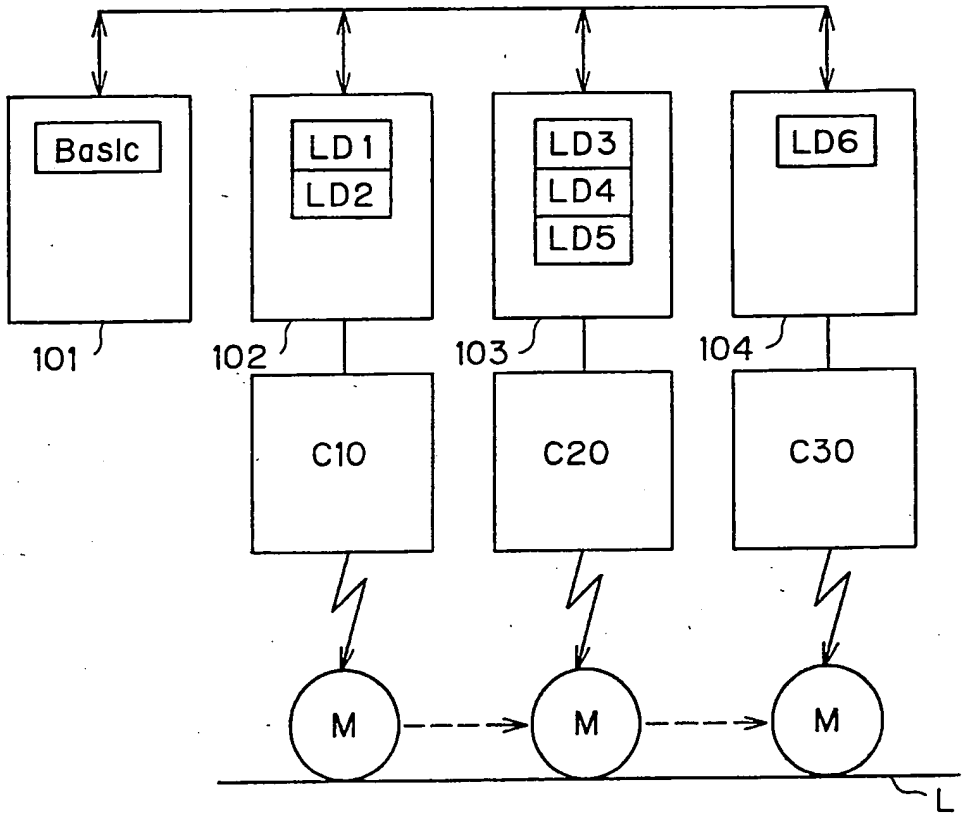


FIG.36

```
10    DO LD1                              LD1
20    DO LD2                              LD2
30    IF COND1 THEN
40       DO LD3                           LD3 or LD4
50    ELSE
60       DO LD4
70    ENDIF
80    PARA
90       DO LD5                           LD5 and LD6
100   AND
110      DO LD6
120   ENDPARA
```

PROGRAMMABLE CONTROLLER

The present invention is directed generally to a programmable controller, into which a sequence control program is incorporated, for inputting and outputting information to a variety of local intelligent appliances in an FA (Factory Automation) field for efficiently controlling factory production lines or in a PA (Process Automation) field for controlling multiple industrial processes.

The present invention is directed more particularly to an improvement of a change-over system associated with a 1-bit processor for executing the sequence control program and an ordinary process. Concomitantly, there are ameliorated a function for a basic process arbitrarily set by the user, a control function over a group of I/O cards for transferring signals to the local appliances and receiving the signals therefrom and a programming workability of the sequence control program set inside.

In general, a CPU board in the programmable controller is equipped with a typical general-purpose processor (e.g., a 16 bit microprocessor 68000 or the like) serving as a first processor and a processor (e.g., a 68000 family or the like) dedicated to execute only an arithmetic operation of numeric values or logic which serves as a second processor. These processors operate while being changed over in accordance with a sequence control program stored in a program memory.

An example of a module, wherein the first and second processors alternately operate to read from the single program memory is described below in relation to Fig. 1.

This type of prior art programmable controller utilises a combination of the first and second proc ssors, resulting in such a troublesome situation in terms of design that the machine language commands stored

in the program memory have to be allocated beforehand to the first and second processors respectively at a stage of system design.

On the other hand, recently oft-utilised commands are interpreter type commands (interpretation program commands) designed independently of the machine language commands intrinsic to the first processor, these interpreter type commands tending to be stored in the program memory.

An apparatus making use of the interpreter type commands is capable of freely setting commands executed by the second processor in addition to the commands executed by the first processor. There is no necessity for allocating the commands to the first and second processors respectively at a stage of designing the first processor. The first or second processor is specified in conformity with contents of the commands read from the program memory, and it is therefore possible to increase a practical efficiency and expand the commands to be executed by the second processor.

In the case of employing the machine language commands and the interpreter type commands, however, the first processor, when the second processor operates, sequentially reads the program commands from the program memory and supplies these commands to the second processor, which in turn leads to a problem of causing a time-consuming transfer of the program.

Where the interpreter commands are used, the first processor has to interpret the readout commands and determine a processor for execution. Especially in an apparatus where the first and second processors are frequently changed over, an excessive period is created.

It is a primary object of the present invention, which obviates the foregoing problems, to provide a programmable controller capable of facilitating an operation of changing over the first and second

processors and speeding up processes as a whole.

In applying the invention, there can be provided a programmable controller intended to improve not only an internal geometry of CPU boards but also efficiencies of arithmetic processing for general purposes, information processing, a control operation, basic program processing associated with communication with a computer system connected to a host side and sequence control program processing.

Also in applying the invention, there can be provided a programmable controller intended to securely transfer data at a high velocity by improving a function to transfer the data to a group of I/O cards, connected to the CPU boards, for transmitting signals to local appliances and receiving the signals therefrom.

Also in applying the invention there can be provided a programmable controller capable of simplifying creation of a sequence control program and also detecting an operating abnormality of the programmable controller itself and an abnormality in advancement of the sequence control program.

In order that the invention may be clearly understood and readily carried into effect examples thereof will now be described in relation to an example of the prior art with reference to the accompanying drawings, in which:

Fig. 1 is a block diagram illustrating a prior art programmable controller;

Fig. 2 is a block diagram showing an embodiment of a programmable controller according to the present invention;

Fig. 3 is a diagram showing a content of a program memory for explaining operations of the apparatus depicted in Fig. 2;

Fig. 4 is a block diagram illustrating another embodim nt of the programmable controller of

the invention;

Figs. 5(a) and 5(b) are diagrams each showing operations of the apparatus depicted in Fig. 4;

Fig. 6 is a block diagram illustrating an apparatus, wherein a construction of the embodiment of Fig. 4 is partly modified;

Figs. 7(a) and 7(b) are diagrams each showing operations of the apparatus depicted in Fig. 6;

Fig. 8 is a block diagram showing still another embodiment of the programmable controller of the invention;

Figs. 9(a) and 9(b) are diagrams showing contents of a first and second program memory regions depicted in Fig. 8;

Figs. 10(a) and 10(b) are block diagrams in combination showing an apparatus in which some components of the embodiment of Fig. 8 are modified;

Fig. 11 is a diagram illustrating a processor board of the programmable controller of the invention and a geometry of an I/O board;

Fig. 12 is a diagram showing a ladder program for explaining operations of a programmable controller of the invention;

Fig. 13 is a diagram of assistance in explaining operations of first and second processors of a programmable controller of the invention;

Fig. 14 is a view depicting a system which employs the programmable controller;

Fig. 15 is a diagram showing a correlation between basic process and a sequence control process of a programmable controller of the invention;

Fig. 16 is a flowchart in the case of setting processing languages from a host computer with respect to a programmable controller of the invention;

Fig. 17 is a conceptual diagram showing operations of a standard I/O driver of a programmable controller of

the invention;

Fig. 18 is a diagram showing a mode of creating a process definition table in the standard I/O driver of a programmable controller of the invention;

Fig. 19 is a flowchart showing I/O board access procedures by the standard I/O driver of a programmable controller of the invention;

Figs. 20 and 21 are time charts each showing a data transfer between an I/O board and a CPU board of a programmable controller of the invention;

Fig. 22 is a diagram illustrating a construction of an abnormality recording module incorporated in a programmable controller of the invention;

Fig. 23 is a diagram showing a case where pieces of information of the abnormality recording module depicted in Fig. 22 are displayed on a CRT of a host computer;

Fig. 24 is a diagram depicting a ladder circuit to which a variety of comments generated by a programming tool are added;

Fig. 25 is a block diagram representing a programming tool of the invention and a function to display the comments corresponding to step numbers;

Fig. 26 is a diagram showing an intrafunctional comment file in relation to Fig. 25;

Fig. 27 is an intrafunctional circuit/comment corresponding table in relation to Fig. 25;

Fig. 28 is a diagram showing transitions of CRT picture display in the programming tool based on the function of Fig. 25;

Fig. 29 is a chart showing a processing routine of a programmable controller of the invention;

Fig. 30 is a chart showing functional blocks at the time of ladder circuit programming in a programmable controller of the invention;

Fig. 31 is a diagram showing a creation of a ladder circuit;

Fig. 32 is a diagram showing correspondence of signal names of the ladder circuit to addresses;

Fig. 33 is a diagram showing allocations of detailed addresses to the signal names of the ladder circuit of Fig. 32;

Fig. 34(a) through 34(c) are diagrams each showing a ladder program created per block;

Fig. 35 is a diagram representing a case where a series of sequence control programs are executed by several programmable controllers; and

Fig. 36 is a diagram showing one example of a basic program for controlling the system of Fig. 35.

In the prior art module depicted in Fig. 1, a first processor CPU1 conceived as a typical general-purpose processor (e.g., the 16-bit microprocessor 68000) and a second processor BPU2 defined as a processor (e.g., the 68000 family) dedicated to execute the arithmetic operation of numeric values or logic are connected via an information bus b to a program memory 3. Note that the second processor BPU2 is in some cases referred to as a 1-bit processor.

The program memory 3 stores machine language commands as program commands for the processor CPU1. The processor CPU1 executes processing by sequentially reading the machine language commands. The processor CPU1, when the readout commands are to be executed by the processor BPU2, gives instructions for an operation and a data preparation to the processor BPU2. Subsequently, the commands read from the program memory 3 are sequentially issued to the processor BPU2 with a view to imparting a right of control execution.

Turning now to Fig. 2, there is illustrated a block diagram of an embodiment of a programmable controller according to the present invention.

A characteristic arrangement of this embodiment is that signals are transmittable and receivable between

processors CPU1 and BPU2 by providing a control line L separately from an information bus b.

The processors CPU1 and BPU2 post each other on starting signals and readout addresses of a program memory 3 by use of the control line L.

Note that the program memory 3 serves to store sequence basic commands for the processor BPU2 as well as the above-mentioned interpreter type commands. More specifically, the sequence control program is composed of the sequence basic commands used for a 1-bit arithmetic operation and created by machine languages and sequence application commands used for a comparison between numeric values and data processing and created in an interpreter format, the sequence control program being stored in the program memory 3 in accordance with the sequence.

The descriptive emphasis will now be placed on operations of such an apparatus with reference to a block diagram of Fig. 3 illustrating an internal geometry of the program memory 3.

Referring to Fig. 3, application commands 1 to 4 may be conceived as commands (interpreter type commands) for the processor CPU1, while basic commands 1 to 4 may be defined as commands (machine languages) for the processor BPU2.

The processor CPU1 behaves to sequentially read the application commands 1 and 2 from the program memory 3 in the order indicated by an arrow A. The processor CPU1 interprets and executes these commands.

Next, CPU1 reads the basic command 1 (an arrow B) of an address X1 and interprets this command. If judged as a command for BPU2, the processor CPU1 transmits the address X1 of the basic commands 1 in the program memory 3 to BPU2 via the control line L and at the same moment starts up the processor BPU2.

As a result, the processor BPU2 obtains a right of

use of the information bus b and then xecutes, as indicated by an arrow C, the basic commands 1 to 4 in accordance with the address X1 of the program memory 3 which is received from CPU1.

At this time, BPU2 does not receive, as in the case of the prior art apparatus, the basic commands via CPU1 but reads by itself the basic commands 1 to 4 directly from the program memory 3.

The basic commands are those intrinsic to the processor BPU2, and the processor BPU2, on the occasion of execution, does not effect the interpretation.

Now, the processor BPU2 reads the application command 3 of an address X2 and judges that this command should be executed by CPU1 (an arrow D). The processor BPU2 in turn transmits, to CPU1 via the control line L, the address X2 of the application command 1 in the program memory 3, thereby starting up CPU1.

In consequence of this process, the processor CPU1 acquires the right of use of the information bus b and thereafter performs the reading process with respect to the address X2 from the program memory 3 and then executes the operation (an arrow E).

The processors CPU1 and BPU2 thus transmit and receive the starting signals and addresses of the program memory 3 via the control line L and alternately operate. Particularly, BPU2 reads the commands not through CPU1 but directly from the program memory 3, thereby making high-speed processing possible, correspondingly.

As discussed above, the processor BPU2 does not require the interpreting operation when executing the basic commands, and this period can thereby be omitted.

Turning next to Fig. 4, there is shown another embodiment of the programmable controller of the invention.

In this embodiment, CPU1 is connected via a first information bus b1 to BPU2 which is in turn connected via

a second information bus b2 to the program memory 3.

The processor BPU2 incorporates a program command take-in circuit 21 connected to the information bus b2 on the side of the program memory 3, a program command readout circuit 22 connected to the information bus b1 on the side of CPU1 and further a pseudo command generating circuit 23 for holding a pseudo command representing "no execution" with respect to CPU1. The program command readout circuit 22 includes a change-over circuit 24 for selecting the program command take-in circuit 21 or the pseudo command generating circuit 23.

In this embodiment also, as is similar to the embodiment of Fig. 2, the program memory 3 stores the interpreter type commands and the BPU2 basic commands as well.

The operations of the thus constructed apparatus of the invention will be explained with reference to Figs. 5(a) and 5(b).

Fig. 5(a) illustrates a case where CPU1 executes the commands.

The processor BPU2 judges that the command 1 read at an address Y1 is catered for CPU1 and changes over the change-over circuit 24 to the program command take-in circuit 21 within a dedicated processor 2. The readout command 1 is imparted from the program command readout circuit 22 via BPU2 to the processor CPU1 which in turn starts operating.

Note that the readout addresses with respect to the program memory 3 are generated from BPU2 even during operation of CPU1.

Turning to Fig. 5(b) there is shown a case where BPU2 executes the commands.

The processor BPU2 judges that the command 2 read at an address Y2 is to be processed by itself, whereby the pseudo command circuit 23 is connected to the pr gram command readout circuit 22 with the aid of the change-

over circuit 24. With this arrangement, BPU2 reads the command from the program memory 3 in accordance with the readout address generated by itself and then executes the command, while CPU1 performs nothing in conformity with a pseudo command N representing "no execution" which is given from BPU2.

In this manner, the commands read from the program memory 3 are surely imparted to CPU1 or BPU2, thereby smoothly effecting change-over between CPU1 and BPU2 without causing futility in time.

In the example shown in Fig. 4, the processor BPU2 is middled between CPU1 and the program memory 3. However, another arrangement equivalent thereto is that CPU1 may be middled therebetween by providing circuitry in CPU1, which corresponds to the program command take-in circuit 21, program command readout circuit 22, pseudo command generating circuit 23 and change-over circuit 24.

In this example, the processor interposed therebetween acts to generate addresses with respect to the program memory 3. Hence, even after changing over the processor which is to execute the command, a situation of advancement of the program is constantly grasped by the middled processor, and it is therefore possible to omit a period for transmitting and receiving the addresses with respect to the program memory 3.

Turning now to Fig. 6, there is given an arrangement in which the addresses generated on the side of CPU1 can be sent to the program memory 3 in the embodiment of Fig. 4.

In this example, the processor BPU2 incorporates an address reading circuit 25 for taking in addresses generated by CPU1 toward the information bus b1, an address transferring circuit 26 for imparting the readout addresses of the program memory 3 to the information bus b2 and a change-over circuit 27. The change-over circuit 27 serves to connect the addr ss transferring circuit 26

to the address reading circuit 25 or to an address generating circuit provided in an interior of the dedicated processor 2.

Operations of this embodiment will hereinafter be described in conjunction with Figs. 7(a) and 7(b).

Fig. 7(a) illustrates a case where CPU1 executes the commands. The change-over circuit 24 incorporated in BPU2 is connected to the program command take-in circuit 21, while the change-over circuit 27 is connected to the address reading circuit 25. An address Z1 generated in CPU1 is transferred via BPU2 to the program memory 3, and the command 1 corresponding thereto passes through BPU2 and is then given to CPU1.

Fig. 7(b) shows a case where BPU2 executes the commands. The change-over circuit 24 incorporated in BPU2 is connected to the pseudo command generating circuit 23, while the change-over circuit 27 is connected to the address generating circuit 28. The processor BPU2 reads and executes the command 2 in accordance with an address Y3 generated inside, whereas CPU1 executes nothing after reading a pseudo command N from BPU2.

The arrangement exemplified in Fig. 6 makes CPU1 accessible to an arbitrary position of the program memory 3 and is also advantageous when executing a jump command or the like.

In accordance with the embodiment shown in Fig. 4, or 6, the processors CPU1 and BPU2 thus operate alternately in conformity with the commands read from the program memory 3. Especially BPU2 reads the commands directly from the program memory 3, and hence high-sped processing is practicable, correspondingly.

Since the basic commands may be defined as those intrinsic to BPU2, there is no necessity for interpretation on the occasion of execution by BPU2, and it is therefor possible to omit this period.

Referring to Fig. 8, there is illustrated another

embodiment of the programmable controller of the present invention.

In this embodiment, CPU1 is connected via the information bus b1 to a first program memory region 31 into which only applications commands are stored. Connected via the information bus b2 to BPU2 is a second program memory region 32 into which only basic commands are stored. A first output port 41 is connected to the information bus b1, while a second output port 42 is connected to the information bus b2. Outputs of the output ports 41 and 42 are imparted to a signal synthesizing unit 43 which in turn gives instructions for initiating or stopping the operation to CPU1 and BPU2.

Figs. 9(a) and 9(b) show contents of the program memory regions 31 and 32.

The program memory region 31 is intended to store application commands 1 to 5 and an application command OUT for instructing the initiation of operation to BPU2, while the program memory 32 is intended to store basic commands 1 to 4 and a basic command OUT for instructing the initiation of operation to CPU1.

The description will next be focused on the operation of this embodiment.

To start with, it is assumed that the processor CPU1 is functioning.

At this time, the signal synthesizing unit 43 issues an operating instruction to CPU1. Whereas to BPU2, there is given an instruction for stopping the operation.

The processor CPU1 sequentially reads the application commands 1 and 2 from the program memory 3 and then executes these commands. Subsequently, CPU1 reads the application command OUT for issuing the operating instruction to BPU2 and transmits this command to the output port 41.

The signal synthesizing unit 43 effects a reading

process from the output port 41 and recognises the application command OUT, at which time CPU1 receives an instruction for stopping the operation while giving an operating starting instruction to BPU2.

As a result, BPU2 in turn initiates the operation and executes the basic command after reading it from the program memory region 32. The basic command 2 is executed, and the basic command OUT is read out. The basic command OUT is transferred to the output port 42.

The signal synthesizing unit 43 performs a reading process from the output port 42 and recognises the basic command OUT. Then, the signal synthesizing unit 43 gives an operation stopping instruction to BPU2 and an operation starting instruction to CPU1.

Consequently, CPU1 resumes its operation.

In this manner, CPU1 and BPU2 are smoothly changed over by means of the signal synthesizing unit 43.

Figs. 10(a) and 10(b) in combination show an example, substantially similar to the embodiment of Fig. 7, wherein the output ports 41 and 42 and the signal synthesizing unit 43 are connected en bloc to one information bus b.

In this example, a configuration of the program memory 3 is that there are, as depicted in Fig. 10(b), set beforehand a region A1 for storing only the application commands and a region A2 for storing only the basic commands.

The operations of this embodiment are absolutely the same as those in the apparatus depicted in Fig. 8.

In the manner discussed above, the processors CPU1 and BPU2 operate alternately in conformity with the commands read from the program memory regions 31 and 32 or from the inter-program-memory regions A1 and A1 in the embodiment shown in Fig. 8 or Fig. 10. In particular, BPU2 reads the dedicat d commands directly therefrom, which permits high-speed processing, correspondingly.

Since the basic commands are classified as those intrinsic to BPU2, the interpretation is not required on the occasion of execution by BPU2, and this period can thereby be omitted.

According to the present invention described above, Fig. 11 depicts an example where the programmable controller is actually constructed by use of each circuitry.

Referring to Fig. 11, CPU1, BPU2 and the program memory 3 are connected to the information bus b in a CPU board 100. To be more specific, CPU1 is a microprocessor, e.g. MC68000 or the like, for controlling the CPU board 100 as a whole. The microprocessor CPU1 is disposed in parallel with BPU2, connected to the information bus b, for executing a sequence basic command (a 1-bit processing command). A CPU gate array 5 is a block for output-controlling timing signals so that CPU1 is allowed to effect the execution at a high efficiency.

BPU2 is constructed of a gate array to process the sequence basic command conceived as a 1-bit processing command at a high velocity, BPU2 being connected via a command bus (not illustrated) directly to the program memo (RAM128KB) for storing a sequence control program such as a ladder program or the like.

Note that the information bus b includes an address bus, a data bus and a control bus.

A variety of commands to be executed by CPU1 for controlling the entire board are stored in a ROM (256KB) 6. A data memory (RAM64KB) 7 is an operating region of CPU1, into which a basic program which will be mentioned later is also stored.

An I/O interface (I/F) 8 is provided for an I/O bus bb connected to I/O boards C1 and C2. The number of I/O boards, though only two boards are connected in the figure, may increase depending on a system configuration.

Connected to the infromation bus b are a realtime clock generating unit 9 having a timer function, a communication buffer (RAM32KB) 10 used for communication and a host interface 11 connected to the communication buffer 10 and serving as an interface to a host bus B. There is also provided an RS232C port 12 for communication with a programming tool.

Connected to the I/0 bus bb are a plurality of I/0 boards of two kinds, i.e., a general register/interface type I/0 board C1 and a command/interface type I/0 board C2, incorporating a microprocessor, for effecting communications by transferring the commands to and receiving them from the host programmable controller.

The register/interface type I/0 board C1 having the microprocessor is composed mainly of an interface c11 to the bus bb and an interface c12 to an outside contact. The I/0 board C2 consists of an interface c21 to the bus bb, a microprocessor c22 (e.g., 8-bit CPU), a data memory c23 (e.g. RAM8KB) and an interface c24 to the outside.

Characteristics of this CPU board 100 will next be explained.

The characteristic in terms of its geometry is that BU2 performs a direct access to the program memory 3 not through the information bus b but through only the command bus (not shown). The functions of the CPU board 100 will be described with reference to Figs. 12 and 13.

Turning to Fig. 12, there is illustrated a ladder program as an example of the sequence control program. Fig. 13 shows a string of program commands corresponding to the ladder program.

Now, when starting the sequence control, CPU1 actuates BPU2 which in turn reads a string of programs from the program memory 3 and then initiates processing from a load command "LD" defined as a sequence basic command.

Subsequent to this process, BPU2 executes an AND

command, an OR command, an OUT command and a LD command. If the readout program commands come under the application command (1) to be executed by CPU1, CPU1 takes over a control executing right including an occupying right of the address bus and the data bus from BPU2. Then, it follows that the application command (1) is executed by CPU1. Upon a completion of processing the application command (1), CPU1 informs BPU2 of the completion thereof, and BPU2 reads the next command. If the next application command (2) should be executed by CPU1, BPU2 hands over the executing right to CPU1 once again.

A program command that BPU2 is to read next is the load command "LD", and hence BPU2 executes this command by itself.

As discussed above, BPU2 which executes the sequence basic command invariably reads a string of program command. When the readout command is classified as a sequence application command to be executed by CPU1, CPU1 takes over the control executing right therefrom. In this case,a great majority of sequence basic commands exist in the string of program commands, and a velocity of processing by BPU2 increases. CPU1, after executing the commands, merely sends back a termination notice to BPU2.

If the program command read by BPU2 is a false command undefined in the system, BPU2 is preset to supply CPU1 with a pseudo command corresponding to a form of the false command. This pseudo command is set with the intention of exerting no adverse influence on the system by causing CPU1 to execute the false command as it is.

When CPU1 starts performing the sequence control process in this manner, BPU2 reads the program command from the program memory 3 and starts executing the command. Thereafter, BPU2 reads the program command directly from the pr gram memory 3, and hence there is no

necessity for CPU1 to read the program commands on by one and for setting the control executing right.

Referring to Fig. 14, there is shown an example where the programmable controller consisting of the CPU board 100 and a plurality of I/0 boards as applied to the field of factory automation.

In the system depicted in Fig. 14, a programmable controller U2 controls a mechanical part machining system S1, while a programmable controller U2 executes control by inputting and outputting signals to a mechanical part assembling system S2. The programmable controllers U1 and U2 communicate via a bus B wth a desk top computer DTC including a printer P, thus informing the operator of a system condition. It is to be noted that the desk top computer DTC is managed by a middle-sized or large-sized host computer MC through a communication line A as the case may be.

More specifically, the programmable controllers U1 and U2 perform sequence control operations such as a contact I/0 process, motor driving/stopping processes, a part positioning process and lamp lighting/extinguishing processes with respect to the systems S1 and S2.

The programmable controller U1 will next be emphasised for explanation.

The programmable controller U1 comprises the CPU board 100 depicted in Fig. 11 and the I/0 boards C1 and C2 fitted in inter-unit slots, the controller U1 being connected to a bus (I/0 bus bb). Additional components are a power supply boards PS for supplying electric power and an extension board EX participating in communications with the other programmable control U2.

Based on such a construction, a sequence control program adaptive to the mechanical part machining system S1 is incorporated into the CPU board 100 of the programmable controller U1, whereby a sequence arithmetic operation is executed on the basis of information giv n

from I/0 board. Control signals are transmitted via the I/0 board to the mechanical part machining system S1, thereby executing a desired sequence operation.

The above-described sequence control program is transmitted to a program creating tool viz., in this example, a ladder program is transmitted to the CPU board 100 in an edit format on a CRT picture of the desk top computer DTC.

The programmable controller U2 has also much the same construction and operations as those of the programmable controller U1.

Now, program processing in the CPU board 100 and processing on the side of the I/0 board will next be described in greater detail.

CPU1 executes the sequence control and at the same moment carries out quite typical operations such as communications of multiple data with a host computer, arithmetic operations for general purposes, information processing and a control operation. A basic program created by using basic languages is set. The description will next deal with a correlation between execution of the basic program and execution of the sequence control program.

Turning to Fig. 15, there is illustrated a conceptual diagram representing parallel operations of a sequence control process SQ and a basic process BAS.

The sequence control process SQ is intended to execute a string of program commands shown in Figs. 12 and 13, while the basic process BAS is associated with a data readout program which is, as stated earlier, arbitrarily created by the user or a basic program like communication program. An execution right change-over processing unit ES, a timer T and a task scheduler TS have functions which are set softwarewise inside the CPU board 100. The task scheduler TS has a function to determine the priority for execution with respect to

several processes in the basic program. This function is not, however, associated directly with the operations of the present invention. Note that generally several hundreds or several thousands of sequence control program strings are provided, and executing time of one cycle is several tens ms or several hundreds ms which largely differs depending on a configuration of the system to be controlled.

It is now assumed that there is prepared a system in which 10 ms is preset in the timer T.

The first step of the sequence control process SQ is to execute a load command "LD". Subsequently, the execution moves to an AND command AND, an OR command OR and an output command OUT, at which time 10 ms passes. Then, a time-up signal as an interruption signal is, it is assumed, transmitted from the timer T to the execution right change-over control unit ES. CPU1 inputs this interruption signal, and at this time the control executing right is still held in BPU2. As a result, CPU1 is unable to receive this time-up interruption.

The sequence process further advances, and BPU2 executes the load command "LD". Next, CPU1 does not receive a 10 ms time-up interruption signal until the control executing right is handed over to CPU1 and a process of the application command (1) is started. Consequently, the execution right change-over processing unit ES changes a process which is to be executed by CPU1 from the sequence process SQ to the basic process BAS, with the result that CPU1 starts processing the basic program BAS in accordance with a task scheduler 83.

At this time, the execution of the sequence control process SQ remains stopped.

Thereafter, the timer T is brought into a time-up state, and the 10 ms interruption signal is generated. At this time, the execution right change-over processing unit ES in CPU1 retreats a present basic advancing

situation temporarily to a data memory 7, and the operation of the sequence control process SQ which is being stopped resumes.

In the case of completing the sequence control process SQ, the sequence control process SQ imparts a notice of completion to the execution right change-over processing unit ES irrespective of the time-up signal per 10 ms, and CPU1 initiates the basic process BAS.

As discussed above, the operation is carried out by changing over the sequence control process SQ and the basic process BAS per 10 ms, with the result that the sequence process SQ and the basic process BAS appear from outside as if they were executed simultaneously. Hence, multi-processing is practicable.

Note that the time set in the timer 82 is not limited to 10 ms but may arbitrarily be set depending on the system configuration.

On the other hand, some of the system constructions do not need the parallel operations of the sequence control process and the basic process. In some cases, either the sequence control process or the basic process is unnecessary depending on modifications of the system. On that occasion, the CPU board 100 previously incorporates a program pursuant to the flowchart of Fig. 16, and the countermeasures are taken as follows.

After executing a self-diagnostic operation when turning On the power, whether or not a process available language is designated from the host side is examined. If the available language is a sequence language, a system table necessary for processing the sequence language is created in the data memory 7. If the basic language is designated, a system table required for processing the basic language is created in the data memory 7. If both of the sequence language and the basic language are designated, system tables necessary for processing both the sequence language and the basic language are created

in the data memory 7.

In this manner, the program is set to create the system tables corresponding to the languages designated, thereby facilitating the designation of the process languages from the host computer, for instance, a desk top computer.

Next, the explanation will emphasise a standard I/0 driver set in the CPU board 100 and capable of corresponding to various types if I/0 boards.

An operation system OS in the CPU board 100 receives an access request a1 for the I/O board from a use program UP such as a basic program set in the CPU board 100 or an interruption request a2 to the CPU board 100 from the I/0 board. The access request a1 from the user program UP prompts an I/0 requst receiving step (1) to start, while the interruption request a2 to the CPU board 100 prompts an interruption request receiving step (2) to start.

The standard I/0 driver SD according to the present invention generates process definition tables TBL1, TBL2,..., TBLn with respect to the individual I/0 boards fitted to the respective slots when starting up the system. When generating the I/0 request receiving process (1) or the interruption request receiving process (2), a preparation for accessing starts referring to the process definition tables TBL.

When initiating the process, there is made a judgement as to whether the I/0 board to be accessed is clasified as a register/interface type or a command/ interface type, and then a main process thereof is executed. Contents of the register/interface type main process and of the command/interface type main process are identicial with those of the conventional processes. The register/interface type I/0 board may be typified as the ordinary I/0 board C1 depicted in Fig. 11. The command/interface type I/0 board is typified as the I/0

board which, as illustrated in Fig. 11, incorporates a microprocessor to transfer and receive commands.

Operations common to the I/O request receiving process (1) and the interruption request receiving process (2) are to preset it as a common process routine and to execute the common process routine regardless of the type of the interface of the I/O board during the execution of common process. Besides, a special process routine is prepared and set for the I/O board requiring a different process from the standard process. At the start-up the special process routine is used in linkage to the standard I/O driver SD by effecting upload from the I/O board. An address of such a special process routine is set in the process definition table TBL when starting up the system.

A mode of generating the process definition tables TBL will next be explained with reference to Fig. 18.

When performing the first I/O request receiving process of the CPU board 100, the process definition tables TBL are generated with respect to the respective I/O boards by reading various kinds of information and boards ID of the I/O boards.

First, the number of channels (the number of ports) and a necessity or unnecessity for outputting are set as parameters from the multiple information of the I/O boards.

If the I/O board concerned is classified as a command/interface type, addresses of the respective command registers are set in the tables, and subsequently addresses of respective buffers are set in the tables. If the special process is needed, a special process routine address is set in the table.

If the I/O board is classified as a register/interface type, respective register addresses are set in the table.

Finally, flags in which the tables have already

been created are turned ON to execute a variety of request processes.

The above-mentioned operations are performed with respect to the respective I/O boards provided in the system, thereby setting the process definition tables TBL in the standard I/O driver SD.

There will next be exemplified accessing of the standard I/O driver SD having the process definition table TBL in conjunction with a flowchart of Fig. 19.

Upon an initiation of accessing i.e., a data transfer preparing process, the standard I/O driver SD of the CPU board 100 refers to the process definition table TBL of the I/O board concerned, checks the number of channels (the number of ports) of the I/O board concerned and presence or non-presence of output and further identifies the type of the interface.

If the classified as a command/interface type, a data conversion is effected by getting an output buffer address while referring further to the process definition table TBL. The data outputted to the output buffer are set. Referring again to the process definition table TBL, an output command is set by getting a command register address, in which state there is a wait for interruption from the I/O board. Immediately when receiving the interruption, a return status is outputted.

If the I/O board concerned is of a register/interface type, whether it is a data I/O interface or not is checked by getting the output data register address while referring to the process definition table TBL. In the case of the data I/O interface, 1-bit data is converted into 1-word data, and mask data is set. Then, the data is written to the output data register. This process of writing the data to the output data register is common.

In order to equip the above-described standard I/O driver SD, the multiple I/O boards are mounted in the

system. Even in such a case, th process definition tables are created for every I/0 board, and accessing to the I/0 boards is carried out on the basis of the process definition tables. Hence, there is no necessity for providing the driver per I/0 board. It is possible to correspond to accessing to all the I/0 boards without causing any decline of performance by using only one standard I/0 driver. A variety of I/0 boards can be employed with the single standard I/0 driver, resulting in zero in number of subsequent steps of developing the I/0 driver. This reduces a capacity of the memory for storing the I/0 driver. Furthermore, the interfaces of the I/0 boards are standardised into the register type or the command type, whereby the user is able to arbitrarily design the I/0 board and incorporate it into the system.

A mode of hardwarewise data transfer to the I/0 board connected via the I/0 bus bb to the CPU board 100 will next be described with reference to Figs. 20 and 21.

Referring to a block diagram of Fig. 11, strobing signal generating means for generating strobing signals STB* on the control line in the I/0 bus bb when the data transfer request arises are provided for an I/0 interface 8 with respect to the I/0 bus bb of the CPU board 100 and for interfaces c11 and c12 of the I/0 boards C1 and C2 respectively. To be specific, logic constitutions for transferring the strobing signals STB* to the I/0 bus bb per transfer frame in association with the data transfer request are added to the I/0 interface 8 composed of a gate array and the interfaces I/Fc11 and I/Fc12.

Fig. 20 is a time chart showing a case where the I/0 board transfers the data to the CPU board 100 in response to the data transfer request of the CPU board 100.

When the data transfer request is generated from the CPU board 100, the CPU board 100 transmits a control frame FO and strobing signals STB*"L" to the I/0 bus bb.

Set in the control frame FO ar bite enable bits for enabling the access on the basis of transfer mode bit bites which perform a read/write identification and specifying data sizes (2 or 4 bites) and types of data (data or control signals). The control frame FO becomes effective at a first transition edge SO of the strobing signal STB*"L".

Subsequently, the CPU board 100 transmits address frames F1 and F2, each composed of 2-bite frame, for securing a 256KB address space. These address frames F1 and F2 become effective at last transition edges S1 and S2 of the strobing signals STB*"L".

A period T1 up to this point covers the operations on the part of the CPU board 100.

The I/O board receiving the frames FO, F1 and F2 judges whether the board itself is selected or not during the period T1. If selected, the I/O board executes a data tansmission preparing process.

The selected I/0 board sends out data frames F3 and F4 to be transmitted onto the I/0 bus bb during a period T3. In this case, the data is 2 bites. Simultaneously, the I/0 board transmits the strobing signals STB*"L". The data frames F3 and F4 become effective at last transition edges S3 and S4 of the strobing signals STB*.

Finally, the I/0 board transmits a status frame F5 together with the stobing signal STB*"L". The 1-bite status frame F5 may be conceived as an inside condition signal of the I/0 board concerned, i.e., a signal for setting a variety of conditions such as a data normal transfer, an error status and a board failure condition.

Note that a frame signal FRM* represents the first frame at its last transition edge and also the last frame at the first transition edge and is utilised for detecting a framing error. A data direction specifying signal DDIR idicates a frame transfer from the CPU board 100 to the I/0 side at the time of "H" during a data

transfer cycle and also a frame transfer from the I/0 board to the CPU board at the time of "L" and is utilised for preventing a data impingement between the CPU board 100 and the I/0 board. The signals FRM* and DDIR* are transmitted onto a control line of the I/0 bus bb.

Turning to Fig. 21, there is shown a time chart when effecting the data transfer from the CPU board 100 to the I/0 board.

Outputted in this case from the CPU board 100 during a period T4 are the control frame FO, the address frames F1 and F2 and the data frames F3 and F4 together with the strobing signals STB*"L". After the I/0 board has carried out the processes required during a period T5, the I/0 board sends back the status frame F5 and the strobing signal STB*"L" during a period T6.

According to the data transfer mode described in conjunction with Figs. 20 and 21, as discussed above, the CPU board 100 or the I/0 board transmits the strobing signals STB*"L" each time the respective frames are transferred, thereby making the concerned frames effective. Hence, the frames are transferred by a synchronous method. In the case of Fig. 20, the transfer of processes for the periods T1 and T3 is based on an asynchronous method in terms of the data transfer cycle as a whole. In the case of Fig. 21, the transfer of processes for the periods T4 and T6 is based on the asynchronous method. Thus, it is feasible to actualise a mixed method of the synchronous transfer method and the asynchronous transfer method.

Fig. 22 shows a function to record a failure history of the programmable controller. The programmable controller includes abnormality detecting circuits such as a power supply cut-off detecting circuit 21 for detecting that the power supply is cut off, a high temperature abnormality detecting circuit 22 for an abnormality in internal temperature and a software

abnormality detecting circuit 23 for detecting runaway of the software executed inside. If an abnormality detecting signal is generated from any one of these abnormality circuits, this detecting signal is inputted to an interruption generating circuit 24, thereby making an interruption in CPU1.

CPU1 receiving the interruption signal initiates an abnormality occurrence interrupting process 26 from a normal process 25.

On the other hand, CPU1 has a timer function TT to relate a type of abnormality caused to a time at which the abnormality is present when starting the abnormality occurrence interrupting process 26. CPU1 then arranges the abnormalities occurred in time series, while an auxiliary storage unit 27 stores these abnormalities in a file format. The auxiliary storage unit 27 involves the use of a data memory 7.

If designated thereafter from the host computer, CPU1 starts a history readout process 28 and reads contents of the auxiliary storage unit 27 for transmitting them to the host side. This arrangement makes it possible to output abnormality occurring conditions to the outside in a table format shown in Fig. 23 by use of a host-side CRT display unit and a printer. It is also feasible to readily analyse details of the abnormalities produced, a time of occurrence and errors and further take countermeasures against the errors.

Subsequently, there will be explained a manner to specify a ladder circuit in which errors appear during execution of the ladder program or a desired ladder circuit in the programmable controller of the invention.

The programmable controller is connected via, e.g., RS232C and RS422 to a program creating tool. Typically, as illustrated in Fig. 24, the ladder program is set in such a format that a variety of comments are added to th respective ladder circuits. More specifically, in the

ladder program there are set a program name "TEST" at the first stage, a title "trial circuit", step numbers of the ladder circuits, a circuit comment "emergent stop circuit" for a plurality of ladder circuits and subcomments "operating conditions, operating states,..." corresponding to every line of the ladder circuit.

The programmable controller internally stores the above-mentioned ladder program to execute the control process. If an abnormality takes place during the sequence control process, however, the programming creation tool reads all the program steps and displays them on the CRT display picture in order to specify the location of failure. While on the other hand, the ladder circuits have thousands of steps, and it is therefore difficult to seek out the specified ladder circuit on the CRT display picture having a display capability limited to several tens of lines. Generally, all the ladder circuits are printed out, and the program list is brought to the locale for the purpose of making the actual circuits correspond to the circuits on the program. This ladder circuit specifying method, however, exhibits quite poor workability.

In the programmable controller of the present invention, the ladder circuits are specified by the following method.

Fig. 25 is a block diagram depicting an arrangement in which a programming tool is connected to the CPU board 100 of the programmable controller. In the CPU board 100, for simplifying the explanation, only CPU1, an RS232C port 10, RAM7 and an information bus b are illustrated. The programming tool connected to the RS232C port 10 of the CPU board 100 includes a display unit PTCRT, a keyboard PTKB, a main memory PTM1 and an auxiliary memory PTM2.

As explained earlier, the ladder circuits are edited in accordance with the sequence control on the

programming tool and the edited results are temporarily stored in a sequence program image file IMF. A ladder program created is transferred back via an RS232 port 111 to RAM106. At this time, a circuit/comment corresponding table CCT in which the ladder circuits correspond to various kinds of comments is also transferred back together with the ladder program. The circuit/comment corresponding table CCT is intended to make comment positions (step numbers) on the circuits correspond to comment positions on a comment file CF which will be mentioned later. By virtue of this circuit/comment corresponding table CCT, it is possible to store the positioning states between the ladder circuits and the comments thereof even if the program is modified.

With this arrangement, CPU1 sets the sequence program SQP transferred and the circuit/comment corresponding table CCT.

Set in the auxiliary storage unit PTM2 on the side of the programming tool is a comment file CF into which the step numbers of the ladder circuits and the contents of comments added to the ladder circuits are stored while making them correspond to each other.

Fig. 26 shows the contents of the comment file CF. Fig. 27 represents the contents of the circuit/comment corresponding table CCT. The comment file CF may be set not in the programming tool but in RAM7 on the side of the CPU board 100. Besides, there may additionally be added a comment file in which to correspond the step numbers to the subcomments.

Now, an operation of seeking out specific locations of the ladder circuits on the programming tool will be described with reference to Fig. 28. The programming tool reads the contents of the comment file CF and the circuit/comment corresponding table CCT makes an arrangement beforehand to indicate subsequent display pictures S1, S2, S3 and S4 on the CRT display unit.

An initial picture of the CRT display unit of the programming tool is a circuit monitor menu picture S1, from which picture a circuit comment display picture S2 is selected. Then, a circuit comment list is displayed in a list format on CRT. From the circuit comment display picture S2, a subcomment display picture S3 is further selected, and thereafter all the subcomments included in the circuit comment are displayed. When a circuit display picture S4 is selected corresponding to the subcomment, the ladder circuit corresponding to this subcomment is displayed.

Namely, in order to specify a certain ladder circuit from the ladder program shown in Fig. 24, a desired circuit comment is selected by displaying a list of the circuit comment, and a specific ladder circuit can be displayed on the CRT display picture by designating a ladder cirduit corresponding to the subcomment included in this circuit comment.

Note that the circuit display picture S4 may be selected from the circuit monitor menu picture S1 or the circuit comment list S2. Page update scrolling and page update monitoring can be effected on the respective display pictures.

As stated above, in the programmable controller of the invention, the comment file and the circuit/comment corrsponding table are set and then read out. A variety of comments added to the ladder circuits are related to the ladder circuits preparatory to hierarchical displaying on CRT. Hence, it is possible to immediately detect a desired ladder circuit.

The arrangement given above is made to attain an easy-to-search ladder circuits when adjusting the circuits or causing an error. The description will next be focused on improvements of creation of the ladder program and of operation during debugging.

Typically, the sequence control process which

adopts the ladder program is executed by a process routine consisting of a common process like a self diagnosis, an I/0 refresh process of I/0 registers on the side of the I/0 board, execution of the set ladder program and a service process for a host appliance.

Based on the programmable controller of the invention, the I/0 refresh process subsequent to the common process is, as illustrated in Fig. 29, omitted to perform programming and debugging without mounting the I/0 board when creating the sequence control program. With this arrangement, the debugging operation can be done in conformity with an instruction from a debugger such as a programming tool or the like without the I/0 board.

Fig. 30 is a conceptual diagram showing a programming function of the ladder program in the programmable controller of the present invention. Respective blocks in the Figure represent software-functional blocks of the programmable controller of the invention.

The individual functional blocks in Fig. 30 work as follows.

A circuit editing function 201 is to edit respective circuit components of the ladder circuit, by which function the programmer describes addresses of the respective circuit components in the form of signal names similar to device names when designing the ladder circuit. A signal defining function 202 serves as a unit for setting a table format for previously setting a correspondence of the addresses to the signal names of the respective circuit components. A compile function 203 works to transmit a program in an executing format to the sequence process unit 205 with reference to the names of signal in the ladder circuit, the addresses given from the signal defining function 202 corresponding thereto and further the signals from an address automatic

generating function 204. An address automatic generating function 204 is defined as a functional block for automatically assigning the detailed addresses to the signal names given from the signal defining function 202.

Procedures for creating the ladder program by utilising these functions will tangibly be described as below.

A ladder circuit depicted in Fig. 31 is generated in cooperation with the programming tool and the circuit editing function 201. At this time, the individual circuit components of a relay unit, an output unit and so on are set in the form of signal names such as SW1 AND COIL1. However, signal names SW1, COIL1, COIL2, IRL1, TIM1, CNT1 and REG1 are arranged to previously correspond to addresses X, X, Y, Y, I, T, C and D in the signal defining function 202, where the symbol X is the address representing an input, Y is an output, I is an internal relay, T is a timer, C is a counter and D is a data register.

The detailed addresses, which are not set to the respective ladder circuits components in the compile function 203, are automatically assigned in the address automatic generating function 204. To be specific, when not the detailed address Xxxx but the address X alone is set to the circuit element SW1, the detailed address X001 is assigned. At this time, the detailed addresses are set sequentially from the smallest of the numbers added to the signal names of the ladder circuit components. The results are shown in Fig. 33. That is, the address X001 is set to the signal name SW1, and the address X002 is set to the signal name SW2.

In accordance with the executing format program obtained by the compile function, debugging, though processing is performed in the sequence control unit, can be carried out by use of only the CPU board 100 without mounting the I/O board, because the I/O refresh process

is, as illustrated in a chart of Fig. 29, showing a process routine, bypassed even if the I/O board is not mounted at the debugging stage. Based on the debugged results the detailed addresses are added as the necessity arises.

Hence, the signal names can automatically correspond to the detailed addresses without being aware of the addresses of the respective ladder circuit components, which in turn facilitates a design of the sequence control program. Debugging can be effected with no I/O board, and the ladder program operations can be confirmed before finishing a design of a relay board which corresponds to the sequence process.

The programmable controller of the invention performs programming with respect to the ladder program having thousands of steps in the following manner by splitting the blocks per step. Figs. 34(a) to 34(c) show programming modes.

Figs. 34(a) to 34(c) represent steps 1, 2 and 5 as a part of the ladder program.

A start command "ACT PROG1.2" and an end command "INACT PROG1.1", which are newly defined in the present invention, are set at the final substep of the step 1. With this arrangement, when a control operation of the step 1 reaches the final substep, a block ladder program PROG1.2 of the step 2 is started by stopping a block ladder program PROG1.1 of the step 1, thereby initiating the control operation of the step 2.

Set at the final substep of the block ladder program PROG1.2 are a stop command "INACT PROG1.2" of the block ladder program PROG1.2 and parallel start commands "2ACT PROG1.2", "ACT PROG2.1" and "ACT PROG3.1" of the steps 3 to 5. These steps 3 to 5 are simultaneously started.

The ends of th steps 3 and 4 are monitored in the step 5. When detecting ceasing of the steps 3, 4 and 5,

the operation returns to the step 1, i.e., the starting step of the sequence control process in response to stop commands "INACT PROG3.1" and "ACT PROG1.1".

The start command "ACT" and the stop commands "INACT" of the ladder program are thus defined, and hence it is possible to effect programming in parallel by splitting a series of thousands of sequence control programs into several blocks. In addition, the ladder programs inform each other of starting and ending, whereby the sequence control operation can be carried out by setting the ladder programs split into the blocks in a plurality of CPU boards.

Turning to Fig. 35, there is shown an example where a control object M on the actual control line L is controlled by combining a CPU board 101 in which the basic program process is set and CPU boards 102 to 104 in which only the ladder program process is set.

Note that ladder programs LD1 and LD2 are set in the CPU board 102 to which an I/O board group C10 is connected; ladder programs LD3 to LD5 are set in the CPU board 103 to which an I/O board group C20 is connected; and a ladder program LD6 is set in the CPU board 104 to which an I/O board group C30 is connected.

Fig. 36 illustrates an example of the basic program set in the CPU board 101.

The ladder programs LD1 to LD6 are defined as a series of sequence control programs with respect to the control object M, thse ladder programs having the foregoing block construction and being programmed independently.

The basic program of the CPU board 101 functions to issue start instructions to the individual ladder program and receive end instructions thereof.

When starting the operation, the CPU board 101 actuates the ladd r program LD2 subsequent to the ladder program LD1 of the CPU board 102. After these programs

have been ended up, there is executed the ladder program LD3 or LD4 of the CPU board 103 in accordance with the sequence processing results at that time. Immediately after finishing the program LD3 or LD4, in the wake of this step the basic program functions to start in parallel both the ladder program LD5 incorporated into the CPU board 103 and the ladder program LD5 incorporated into the CPU board 104.

On the basis of the illustrated system, the sequence control programs in which a series of sequence control operations are divided into blocks are handled and processed by a plurality of programmable controllers. Thus, highly efficient sequence control processing can be done.

As discussed above, the programmable controller of the invention is capable of improving a velocity of sequence control processing and attaining an easy-to redesign system when being modified. The programmable controller exhibiting a high processing efficiency can thus be actualised.

Although the illustrative embodiments of the present invention have been described in detail with reference to the accompanying drawings, it is to be understood that the present invention is not limited to those precise embodients. Various changes or modifications may be effected therein by one skilled in the art without departing from the scope of the following claims.

CLAIMS:

1. A programmable controller comprising: a first processor; a second processor; a program memory for storing an interpreter type program command and a basic command; an information bus through which said first and second processors and said program memory are connected to each other; and a control line, interposed between said first and second processors, through which bidirectional transmissions of signals are carried out between said first and second processors, characterised in that when said one processor which is on the execution judges that said command be executed by said other processor, said one processor informs said other processor of an address of said program memory to be executed via said control line and effects actuation, and said processor which is to execute said basic command reads said basic command directly from said program memory and executes said basic command.

2. A programmable controller comprising: a first processor; a second processor; a program memory; a first information bus through which said first processor is connected to said program memory for storing an interpreter type program command and a basic command; and a second information bus through which said second processor is connected to said first processor, characterised in that when said first processor itself operates, said first processor reads said program command from said program memory preparatory to execution of said program command and at the same time issues a pseudo command to said second processor, and when said first processor judges that said program command be executed by said second processor, said program command is supplied in place of said pseudo command.

3. A programmable controller comprising: a first processor; a second processor; a first program memory region for storing a program command executed by said

first processor and a command for instructing the execution to said second processor; a second progam memory region for storing a program command executed by said second processor and a command for instructing the execution to said first processor; and a signal synthesising means for inputting said command for instructing the execution to said second processor and said command for instructing the execution to said first processor, and for instructing a start of operation to said one processor and instructing a stop of operation to said other processor.

4. A programmable controller comprising: a group of I/0 boards for transferring and receiving multiple information with respect to a control object; and a processor board for imparting a control signal to said control object via said I/0 board group, said processor board including: a processor for controlling the whole and executing a part of commands in a sequence control program and a basic program for performing general-purpose arithmetic processing, information processing or a control operation by starting said sequence control program and making an end instruction; a 1-bit processor, connected directly to a program memory into which said sequence control program is stored, for executing commands sequentially read from said program memory and imparting said commands to said processor if said readout commands are sequence application commands to be executed by said processor; a data memory for temporarily storing data; a fixed memory for storing a self-diagnostic program; a communication interface participating in a communicating operation with a host computer; an I/0 interface for joining said control object to an I/0 bus through which I/0 boards for transferring receiving multiple information are connected; and an internal bus for mutually connecting said processor, said 1-bit proc ssor, said data memory, said fixed memory, said

communication interface and said I/O interface.

5. The programmable controller as set forth in Claim 4, characterised in that an I/O driver set in said processor board at the starting-up time sets a process definition table for storage in a table format by reading a board ID, a type of interface of said board, the number of channels, a command register address and a buffer address or a data register address and an address for designating a special process when the necessity for said special process arises, and refers to said process definition table when effecting a data outputting process.

6. The programmable controller as set forth in Claim 4, characterised in that: strobing signal generating means are provided in said processor board and said I/O board group; when starting a data transfer cycle, said board for requesting a data transfer transmits strobing signals per frame transmission to make effective said frame concerned; said board receiving said data transfer request transmits data a frame and a status frame or said strobing signals per status frame transmission; and said data transfer cycle is thus ended.

7. The programmable controller as set forth in Claim 4, characterised in that a corresponding number of abnormality detecting circuits to the number of kinds of abnormalities are provided, and said processor receives an abnormality detecting signal as an interruption signal and stores time data given by an internal timer function and a content of the detected abnormality in a table format.

8. The programmable controller as set forth in Claim 4, characterised in that an I/O refresh process in a sequence control process routine is omitted, signal names are assigned to said circuit elements when creating said ladder circuits and made to correspond to addresses in accordance with a preset signal name-to-address

correspondence table, and detailed address s corresponding to said signal names are set sequentially from said address marked with the smallest number to said signal names to which said detailed addresses are not yet assigned at the time of compile.

9. The programmable controller as set forth in Claim 4, characterised in that a series of sequence control operations are split into blocks corresponding to several steps when programming said ladder program, a command for specifying said block to be executed next and a command for specifying a stop of process of said block concerned are set in said ladder circuit of a final substep of each said block, and said ladder program is set and executed per block.

10. The programmable controller as set forth in Claim 4, characterised in that there are set a comment file for storing comments added to said ladder circuits in said ladder program generated in a ladder language and step numbers of said ladder circuit concerned by making said comments correspond to said step numbers and also a circuit/comment table in which said step numbers of said ladder circuit concerned correspond to comment numbers in said comment file, and said circuit comments, said step numbers and said ladder circuits are read from a programming tool.

11. A programmable controller for performing a CPU process by changing over periodically executed sequence processes and a variety of basic processes, comprising: a timer, in which the maximum allowable value of a processing time of said sequence process, for counting a processing time of said sequence process each time; and an executing right change-over processing unit for setting a value of said processing time in said timer by determining said processing time of said basic process in accordance with a count value of said timer and changing over said sequence process and said basic process.

12. A programmabl controller comprising: a singl CPU operating in accordance with a designated languag ; elements required for operating said CPU; a setting means capable of setting at least one or more languages to be executed from outside; and a creating means for creating an optimal executing environment in accordance with said language set by said setting means.

13. A programmable controller comprising: a processor board for processing a basic language; at least one or more processor boards, dedicated to sequence language processing, for storing one or more block-based sequence control programs each dedicated to said sequence language processing; and a gorup of I/0 boards, provided on the lower side of said processor board dedicated to said sequence language processing, for transferring and receiving control information with respect to a control object, characterised in that said processor board for processing said basic language informs each of said block-based sequence control programs in said processor board dedicated to said sequence language processing of a start instruction and an end instruction.

14. A programmable controller substantially as hereinbefore described with reference to Figs. 2 and 3 of the accompanying drawings.

15. A programmable controller substantially as hereinbefore described with reference to Figs. 4 to 7(b) of the accompanying drawings.

16. A programmable controller substantially as hereinbefore described with reference to Figs. 8 to 10(b) of the accompanying drawings.

17. A programmable controller substantially as hereinbefore described with reference to Figs. 11 to 36 of the accompanying drawings.

Published 1990 at The Patent Office, State House, 66-71 High Holborn, London WC1R 4TP. Further copies may be obtained from The Patent Office. Sales Branch, St Mary Cray, Orpington, Kent BR5 3RD. Printed by Multiplex techniques ltd, St Mary Cray, Kent, Con. 1/87